APR 29 1989

NAS1.55:3029

MF-0830-H-10

330

NASA Conference Publication 3029

Seventeenth NASTRAN® Users' Colloquium

Proceedings of a colloquium held in
San Antonio, Texas
April 24-28, 1989

NASA

VL

COMPLETED

NASA Conference Publication 3029

Seventeenth NASTRAN® Users' Colloquium

Computer Software Management and Information Center
University of Georgia
Athens, Georgia

Proceedings of a colloquium held in
San Antonio, Texas
April 24-28, 1989

National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Division

1989

BLANK PAGE

FOREWORD

NASTRAN® (NASA STRUCTURAL ANALYSIS) is a large, comprehensive, nonproprietary, general purpose finite element computer code for structural analysis which was developed under NASA sponsorship and became available to the public in late 1970. It can be obtained through COSMIC® (Computer Software Management and Information Center), Athens, Georgia, and is widely used by NASA, other government agencies, and industry.

NASA currently provides continuing maintenance of NASTRAN through COSMIC. Because of the widespread interest in NASTRAN, and finite element methods in general, the Seventeenth NASTRAN Users' Colloquium was organized and held at The Menger Hotel, San Antonio, Texas on April 24-28, 1989. (Papers from previous colloquia held in 1971, 1972, 1973, 1975, 1976, 1977, 1978, 1979, 1980, 1982, 1983, 1984, 1985, 1986, 1987 and 1988 are published in NASA Technical Memorandums X-2378, X-2637, X-2893, X-3278, X-3428, and NASA Conference Publications 2018, 2062, 2131, 2151, 2249, 2284, 2328, 2373, 2419, 2481 and 2505.) The Seventeenth Colloquium provides some comprehensive general papers on the application of finite element methods in engineering, comparisons with other approaches, unique applications, pre- and post-processing or auxiliary programs, and new methods of analysis with NASTRAN.

Individuals actively engaged in the use of finite elements or NASTRAN were invited to prepare papers for presentation at the Colloquium. These papers are included in this volume. No editorial review was provided by NASA or COSMIC; however, detailed instructions were provided each author to achieve reasonably consistent paper format and content. The opinions and data presented are the sole responsibility of the authors and their respective organizations.

NASTRAN® and COSMIC® are registered trademarks of the National Aeronautics and Space Administration.

BLANK PAGE

CONTENTS

Page

FOREWORD iii

1. IMPROVED PERFORMANCE IN NASTRAN 1
by Gordon C. Chan
(UNISYS Corporation)

2. USING PATRAN AND SUPERTAB AS PRE- AND POSTPROCESSORS TO COSMIC/NASTRAN 91
by Robert R. Lipman
(David Taylor Research Center)

3. INCREASING MARKETABILITY AND PROFITABILITY OF PRODUCT LINE THRU PATRAN & NASTRAN 102
by Art Hyatt
(Deutsch Metal Components)

4. THE USE OF COSMIC NASTRAN IN AN INTEGRATED CONCEPTUAL DESIGN ENVIRONMENT 130
by Gil White
(Intergraph Corporation)

5. A NASTRAN/TREETOPS SOLUTION TO A FLEXIBLE, MULTI-BODY DYNAMICS AND CONTROLS PROBLEM ON A UNIX WORKSTATION 133
by Javier E. Benavente and Norris R. Luce
(Dynacs Engineering Co., Inc.)

6. ENHANCEMENTS TO THE IBM VERSION OF COSMIC/NASTRAN 159
by W. Keith Brown
(RPK Corporation)

7. A POWERFUL ENHANCEMENT TO THE DMAP ALTER CAPABILITY 169
by P. R. Pamidi
(RPK Corporation)

8. A NASTRAN DMAP ALTER FOR LINEAR BUCKLING ANALYSIS UNDER DYNAMIC LOADING 187
by Robert A. Aiello and Joseph E. Grady
(NASA/Lewis Research Center)

9. IMPROVEMENTS TO THE CONTINUE FEATURE IN TRANSIENT ANALYSIS . . . 201
by P. R. Pamidi
(RPK Corporation)

10. FINITE ELEMENT MODELING OF ELECTROMAGNETIC FIELDS AND WAVES USING NASTRAN 214
by E. Thomas Moyer, Jr.
(George Washington University)
and Erwin Schroeder
(David Taylor Research Center)

CONTENTS
(Continued)

Page

11. CALCULATION OF LOW FREQUENCY VIBRATIONAL RESONANCES OF SUBMERGED STRUCTURES 247
by Gordon C. Everstine
(David Taylor Research Center)

12. POWER FLOWS AND MECHANICAL INTENSITIES IN STRUCTURAL FINITE ELEMENT ANALYSIS 262
by Stephen A. Hambric
(David Taylor Research Center)

12. FINITE ELEMENT ANALYSIS OF A MICROMECHANICAL DEFORMABLE MIRROR DEVICE 290
by T. J. Sheerer, W. E. Nelson and L. J. Hornbeck
(Texas Instruments Incorporated)

13. RITZ METHOD FOR TRANSIENT RESPONSE IN SYSTEMS HAVING UNSYMMETRIC STIFFNESS 311
by Thomas G. Butler
(Butler Analyses)

14. MODAL STRAIN ENERGIES IN COSMIC NASTRAN 363
by B. D. Snyder and V. B. Venkayya
(WPAFB, Flight Dynamics Laboratory)

15. DEFINITION OF NASTRAN SETS BY USE OF PARAMETRIC GEOMETRY 382
by Terry V. Baughn and Mehran Tiv
(Southern Methodist University)

IMPROVED PERFORMANCE IN NASTRAN®

by

Gordon C. Chan
UNISYS Corporation
Huntsville, Alabama

SUMMARY

PART I, SPEED IMPROVEMENTS

Three areas of improvement in COSMIC/NASTRAN, 1989 release, were incorporated recently that make the analysis program run faster on large problems, particularly on the VAX computer.

The first improvement on Bulk Data input cards was presented in great detail in the last NASTRAN Users' Colloquium. This paper completes the previous presentation by compiling actual log files and actual timings on a few test samples that were run on IBM, CDC, VAX, and CRAY computers. The tabulated result shows a speed improvement in COSMIC/NASTRAN Link 1 from 3 to 4 times faster on the CDC computer to 50 to 60 times faster on the VAX. IBM and CRAY show 20 to 30 times faster. The speed improvement is proportional to the problem size and number of continuation cards.

The second improvement involves vectorizing certain operations in BANDIT, that makes BANDIT run twice as fast in some large problems using structural elements with many node points (such as 8-, 20-, and 32-node elements). BANDIT is a built-in NASTRAN processor that optimizes the structural matrix bandwidth.

The third improvement is on the VAX computer only. The VAX matrix packing routine BLDPK was modified so that it is now packing a column of a matrix 3 to 9 times faster. The companion unpack routine INTPK is also running 3 to 9 times faster than before. The denser and bigger the matrix, the greater is the speed improvement. This improvement makes a host of routines and modules that involve matrix operation, such as MPYAD, DECOMP/SDCOMP, EMG, REIG/CEIG, PARTN/MERGE, ELIM, SSG2B/C, FACTOR, TRNSP, SDR2, etc., run significantly faster. This improvement also saves disc space for dense matrices; up to two-thirds of the hardware storage space could be saved.

PART II, IMPROVEMENTS, NEW ENHANCEMENTS, AND NEW VERSION

1. A new UNIX version, converted from 1988 COSMIC NASTRAN, was tested successfully on a Silicon Graphics computer using the UNIX V Operating System, with Berkeley 4.3 Extensions. This version with small changes was also known to run successfully on the SUN computer and Apollo workstations.

2. The Utility Modules INPUTT5 and OUTPUT5, developed in 1988, were expanded to handle table data, as well as matrices. INPUTT5 and OUTPUT5 are

general input/output modules that read and write FORTRAN files with or without format.

3. More user informative messages are echoed from PARAMR, PARAMD, and SCALAR modules to ensure proper data values and data types being handled. The ADD module was expanded to handle both single and double precision scale factors.

4. Two new Utility Modules, GINOFILE and DATABASE, were written for the 1989 release. They can be requested via DMAP Alter.

(a) GINOFILE captures any scratch file of the preceding DMAP module and makes it a legitimate GINO file.

(b) DATABASE copies (that is FORTRAN written) the grid point data, element connectivity data, displacement, velocity, and acceleration vectors, loads, grid point forces, eigenvectors, element stresses, and element forces to a user tape (one of the UT1, UT2, INPT, INP1,...,INP9 tape), formatted or unformatted. The grid point data is in system basic coordinates. The displacement vectors can be in system basic or system global coordinates.

5. Seven new rigid elements are added to COSMIC NASTRAN. They are: CRROD, CRBAR, CRTRPLT, CRBE1, CRBE2, CRBE3, and CRSPLINE.

PART I, SPEED IMPROVEMENTS, 1989 NASTRAN

A. Link 1 Improvement

In the sixteenth NASTRAN Users' Colloquium, the author presented a paper entitled "On Bulk Data Cards Processing", in which a new method of processing the NASTRAN raw input data was discussed in great detail. The only thing that was missing in that paper was some actual numbers showing the timing improvement of the 1988 NASTRAN version. Since then, further NASTRAN software improvement, not directly involving the bulk data cards processing, also makes Link 1 run faster. Appendix A tabulates a series of test runs showing the actual timings on several computers, using NASTRAN 87, 88, and 88.5 (which is now 89) versions. Appendix A begins with a cantilever model used throughout all the tests. The model includes 10,000 grid points and 9999 QUAD2 elements, with the grid points and elements intentionally not in sorted order. The actual timings of the XCSA, IFP1, XSORT, and IFP modules were tabulated directly from the NASTRAN log files of various NASTRAN versions on different computers, the NASTRAN new sorter, XSORT2, and the old sorter, XSORT. The conclusion from these tests indicates generally that the new Link 1 is 20 to 30 times faster, with only one exception: CDC tests showed only 2 to 3 times faster. This CDC slowness is explained in the next two paragraphs. The Link 1 improvement on the VAX computer was actually more impressive; it showed 30 to 70 times faster. The new speed improvement of Link 1, however, must be discounted somewhat due to certain source code standardization implemented in the 1987 NASTRAN version. The 1987 version was 3 to 10 times slower then the previous versions.

The CDC computer (a non-virtual memory model) is intended for number crunching. It is, however, extremely slow in handling characters. The new XSORT2 routine, where the bulk data are handled and sorted, and its supporting subroutines are heavily character oriented (that makes this group of routines completely machine dependent). In the 88.5 version, additional modifications to this group of routines were made to avoid CDC's weakness. The actual amount of changes was not too extensive, and therefore will not be a big burden to the other non-CDC computers. The following tests illustrate the CDC deficiency.

A FORTRAN (FTN5) source code 'A=B' is 5 times slower if A and B are characters than if A and B are integers. Similarly, 'If (A .EQ. B) C=D' is 15 times slower if A, B, C, and D are characters than if they are integers. These time-tests were made on the Langley Research Center Y computer (CDC CYBER 185).

B. Vectorizing BANDIT Operation

The grid point connectivity increases exponentially when elements of many node points (8-, 20-, and 32-node elements) are used in a finite element model. Consequently, BANDIT, the NASTRAN built-in bandwidth processor, would require substantial computer CPU time for optimizing the connectivity of the structural model. By modifying the source code using the array vectorizing concept, the BANDIT timing on an actual huge problem was reduced by half.

C. Pack and Unpack on the VAX Computer

All matrix data in NASTRAN are packed; that is, all elements of zero values are squeezed out, when the matrix data are written out to a storage disc. Conversely, the matrix is unpacked into memory space when it is brought back from the disc; that is, all missing zero elements are put back into their correct positions in the matrix. For efficiency, the packing and unpacking routines in NASTRAN are written in the machine dependent assembly languages for IBM, CDC, and UNIVAC computers. However, FORTRAN is used in the VAX. The FORTRAN source code of the VAX, which is much easier to understand, has been studied thoroughly; and improvements were incorporated in key areas. The final result shows great improvement in speed and reduction of disc space, particularly in large and dense matrices. These improvements in matrix packing and unpacking make a host of other modules, such as EMG, MPYAD, SDR2, SMP1, PARTN/MERGE, ELIM, SSG2B/C, FACTOR, TRNSP, SDR2, DECOMP/SDCOMP, EIGR/EIGC, etc. run significantly faster. As indicated by the test runs in Appendix A, all VAX runs on the 88.5 version were at least 30 to 40 percent faster than the corresponding runs on the 88 version.

PART II, IMPROVEMENTS, NEW ENHANCEMENTS, AND NEW VERSION, 1989 NASTRAN

A. UNIX Version of COSMIC NASTRAN

The present trend in computer applications is towards the UNIX operating system. The 1988 COSMIC NASTRAN has been converted to run on a Silicon Graphics computer with UNIX V OS, Berkeley 4.3 Extensions. All the machine independent source code was converted without any major problems. (Only the symbol '4H/*(something)(something)' needed to be changed.) The VAX machine dependent source code (all written in FORTRAN) was used in the UNIX version. Approximately 30 percent of this group of routines required modification and special attention. The major problem encountered here was the usage of the open core. This problem appeared to be very much machine dependent even in the supposedly machine-independent UNIX operation environment. The UNIX versions were also tried on the SUN computer and Apollo workstations successfully.

B. INPUTT5 and OUTPUT5 Improved Capabilities

Utility modules INPUTT5 and OUTPUT5 were incorporated into COSMIC NASTRAN since 1987. OUTPUT5 creates user written FORTRAN files, formatted or unformatted, and INPUTT5 reads the FORTRAN files, also formatted or unformatted. The 1987 versions of INPUTT5 and OUTPUT5 actually handle only GINO (NASTRAN General Input/Output) matrix data blocks. Since matrix data are either real or complex, in single or double precision forms, it is relatively easy to read the GINO matrix data blocks and send them out (OUTPUT5) under FORTRAN control, formatted or unformatted. Similarly, it is quite easy to read (INPUTT5) from a FORTRAN file, formatted or unformatted, and re-create the GINO matrix data block. The 1987 INPUTT5 and OUTPUT5 modules do not handle GINO table data blocks, because a table array normally contains mixed types of data, integers, reals, and BCD data, in endless combinations. To write an array of mixed type data formatted is not an easy task.

The INPUTT5 and OUTPUT5 modules in the 1989 release are expanded to include the GINO table data blocks, as well as the matrix data blocks. Again, the data transfer between the GINO file and FORTRAN file can be formatted and unformatted. The formatted file, in particular, can be used across different computer manufacturers' processors. Appendices B and C provide detailed descriptions of the INPUTT5 and OUTPUT5 modules.

C. Improved and Expanded Data Handling in PARAMR, PARAMD, PARAML, SCALAR, and ADD Modules

The original PARAMR, PARAMD, and SCALAR modules are difficult to use. In most cases, the user has no idea whether the input data he specified is used correctly, and the output from these modules is correctly computed. Unless these modules are further checked by PRTPARM, the user is completely in the dark. Usually, a user has no control over these modules even when errors are found in the input or output parameters.

The PARAMR, PARAMD, and SCALAR modules in the 1989 version provide much more user information. The actual input and output parameters are echoed out. For example, if data are to be abstracted from matrix data blocks by PARAMR/D, the precise row and column positions of the data element are printed, and the user is also informed that the matrix type is real, complex, single precision, or double precision. Similar useful information is echoed out for the table data blocks. In PARAMR and PARAMD, the user is also given the ability to stop or to continue a NASTRAN job at the end of these modules, in the cases of error, or no error found. Appendices D and E provide detailed descriptions of the PARAMD and SCALAR functional modules. PARAML was expanded in the 1988 version. Its manual update is reprinted in Appendix F.

The ADD module, which adds two matrices, [X] = a[A] + b[B], has been expanded. The scale factors a and b can now be in single precision or in double precision. The manual pages for ADD are updated and presented in Appendix G.

D. Two New Utility Modules - GINOFILE and DATABASE

D.1 GINOFILE Module - to Capture A Scratch File of Preceding DMAP Module

NASTRAN's General Input and Output (GINO) utility processor provides three kinds of files for each NASTRAN DMAP operation - input files, output files, and scratch files. Each DMAP module specifies its input files and output files explicitly, and they are saved in the GINO system until an explicit PURGE command is given. The scratch files, however, are not saved, and are left in the computer system unprotected at the end of a DMAP module operation. At this point they are ready to be over-written by anything in the next module. There would be a tremendous amount of work in DMAP (or Rigid Format) programming, and FORTRAN source code changes, if one would like to save one or more of these scratch files by the regular GINO and DMAP rules. After consulting the NASTRAN Theoretical manual and Programmer manual, and possibly studying the NASTRAN FORTRAN source code carefully, some advanced NASTRAN users would like to salvage certain scratch file(s) in a particular DMAP module, so that they can do extraordinary work with those files and DMAP Alters. A new DMAP functional module, GINOFILE, is provided in the 1989 version to capture one scratch file of the preceding DMAP module, and give it a legitimate GINO data block name, and proper GINO data block formation. For example, the scratch file captured may not have a header record and GINO name; the new output file from GINOFILE will have a GINO name and proper header record. This new GINOFILE module will work properly if and only if the scratch file of the preceding module still exists in the computer system, and only the last data written to that file is available if looping is involved. Appendix H describes in detail the usage of GINOFILE module and its limitations.

D.2 DATABASE Module - to Copy Grid Point Data, Element Connectivity Data, and Displacement Vectors (Velocities, Accelerations, Loads, Grid Point Forces, Eigenvectors, Element Stresses, or Element Forces) to a User Tape

Many users have expressed the need to transfer NASTRAN basic data (grids, elements, and displacements, etc.) to an external FORTRAN tape, or file, so that they can use the data for other purposes. Typically, they would like to link up the NASTRAN data to another commercially available program for plotting interactively, or to user written software for data manipulation. Very commonly, the users would like the grid point data and the displacement data in the basic rectangular coordinate system, and all the grid point ID's in their external numbers. To do just that, the users must copy grid point data in the GEOM1 (Geometry 1), or BGPDT (Basic Grid Point Definition Table) file, the Coordinate System Transformation Matrices in CSTM file, element data in the GEOM2 (Geometry 2), or EST (Element Summary Table) file, element type data in the FORTRAN source code GPTABD BLOCK DATA, the EQuivalence EXternal grid point vs. INternal grid point number tables in EQEXIN, and one or two displacement OFP files (Output File Processor, a group of 20 to 25 files). Since all the NASTRAN files are GINO (NASTRAN General Input and Output file processor) written, the users most likely would use the OUTPUT2 module (OUTPUT5 is now available in 88 and 89 releases) for the data transfer. Only then, the users would have enough data to convert the grid points from NASTRAN global coordinates to the basic rectangular coordinate system, and from NASTRAN internal grid point numbering system to the external numbers. Normally, The users have to provide the conversion program either from a commercial source or from their own written programs. In the latter case, the users must be familiar with OUTPUT2 operations, and the contents of various files and their exact data arrangements; they also must have a good knowledge of coordinate transformation, and the User's and the Programmer's Manuals. A seemingly easy job could turn out to be a mammoth task. It is to this end that a new utility module, DATABASE, was written for the 1989 release.

The new DATABASE module copies the grid point data, the element connectivity data, the displacement vectors, and other data out to a user tape (UT1 or UT2 for CDC computer, INPT, INP1, INP2,... INP9 for other computers), formatted or unformatted. All grid points are in basic rectangular coordinates, and the displacement vectors (and others) are in basic rectangular coordinates or NASTRAN global coordinates. All grid point ID's are in their external numbering system. The displacement vectors (and other data) can be real or complex, 'SORT1' or 'SORT2' formats, and single case with one output record, or sub-cases with multiple records. The unformatted tape from the DATABASE module is more efficient. The formatted tape can be printed out for verification, or edited by the system editor. The formatted tape can also be generated on one computer, and used on another computer of a different manufacturer. Appendix I, the user manual update pages for the DATABASE module, shows in detail the records being generated on the output tape. Appendix J presents a NASTRAN example run using this new DATABASE module. The formatted INP1 file of this example run is also listed. Appendix J also includes a FORTRAN program which was used to check out the unformatted tape during the development stage. This FORTRAN program is very useful as a guide to read a typical unformatted FORTRAN tape.

E. New COSMIC Rigid Elements

In addition to the four rigid elements, CRIGD1, CRIGD2, CRIGD3, and CRIGDR, in NASTRAN, seven new rigid elements are now available to COSMIC NASTRAN users.

CRROD - a rigid pin-ended rod element (similar to CRIGDR)
CRBAR - a rigid bar element
CRTRPLT - a rigid triangular plate element
CRBE1 - a general rigid body connected to an arbitrary number of grid points (similar to CRIGD3)
CRBE2 - a rigid body with independent d.o.f. at a single grid point, and with dependent d.o.f. at an arbitrary number of grid points (similar to CRIGD2)
CRBE3 - a rigid body that defines the motion at "reference" grid

points as the weighted average of the motions at a set of other grid points

CRSPLINE- a rigid element of multi-point constraints for interpolation of displacements at specified grid points

The implementation of this group of new elements is as follows.

The input data of CRROD, CRBE2, and CRBE1 are mapped into CRIGDR, CRIGD2, and CRIGD3 data formats respectively, and thus they are treated as a CRIGDR, a CRIGD2, or a CRIGD3 element.

The input data of CRBAR and CRTRPLT are mapped into the rigid general element CRIGD3 data format, and they are each treated as a CRIGD3 element.

The CRBE3 and CRSPLINE have no COSMIC old rigid element equivalence. A special subroutine was written to handle these two elements.

Appendix K provides the Users' Manual update pages for this new group of rigid elements.

F. Other Improvements

Other improvements in the 1989 release include:

(a) A new 'ECHO = NONO' option that provides absolutely no input card (and restart) echo.

(b) DIAG 38 will list each element being processed by the EMG module. Thus it may pinpoint which element that might have an input error.

(c) Complex stresses and forces for QUAD4 elements.

(d) Inclusion of QUAD4 in random analysis.

APPENDIX A

NASTRAN LINK1 TIME TESTS

(A) NASTRAN TIMING TEST INPUT DECK

```
NASTRAN BULKDATA=-1, TITLEOPT=-1
ID       TEN THOUSAND GRID POINTS TEST PROBLEM, CM=250,000 (CDC)
$                                               50K  WORDS (VAX)
$                                               7.5M BYTES (IBM)
$                                               100K WORDS (UNIVAC)
$                                               OPEN CORE? (CRAY)
APP      DISP
SOL      1,0
$DIAG    42    $ TO USE OLD XSORT MODULE
TIME     20
CEND
TITLE    = DATA INTENTIONALLY GENERATED NOT IN SORTED ORDER
SUBTITLE = NO CONTINUATION CARDS IN DECK
DISP     = ALL
ECHO     = NONE   $ OR  ECHO = SORT
SPC      = 135
LOAD     = 246
BEGIN BULK
$
$              TEST MODEL - A LONG CANTILEVER BEAM
$   Y
$   |
$   |
$ 5001  5002  5003  5004  5005  5006  5007 .. 9998  9999  10000 = GRID PTS
$   +-----+-----+-----+-----+-----+-----+--    --+-----+-----+
$   |     |     |     |     |     |     |        | (49 | (49 |  (QUAD2 ELEM.ID
$   | (1) | (2) | (3) | (4) | (5) | (6) |  ...   | 98) | 99) |   IN BRACKETS)
$   +-----+-----+-----+-----+-----+-----+--    --+-----+-----+ -------------- X
$   1     2     3     4     5     6     7     4998  4999   5000
$
GRID, 1,, 0.0,    0.0,  0.0
=(2499), *(2),, *(2.),==
GRID, 5000,, 4999.0,   0.0,  0.0
=,*(-2),,*(-2.),==
=(2498)
CQUAD2, 1, 10,  1,   2, 5002,  5001
=(2498),*(2),=,*(2),///
GRID, 5001,, 0.0,  1.0,  0.0
=,*(2),,*(2.),==
=(2499)
GRID, 10000,, 4999.0  1.0  0.0
=,*(-2),,*(-2.),==
=(2498)
PQUAD2, 10   20      0.02
MAT1,   20   30.+6,, 0.33
CQUAD2, 4998  10  4999  5000   10000  9999
=,*(-2),=, *(-2),*(-2),*(-2),*(-2)
=(2497)
SPC1, 135  123456  1  5001
FORCE,246, 5000,, 200., 0. 0. -1.
=,=,      10000,==
ENDDATA
```

(B) TO GENERATE A FIXED-FIELD INPUT DECK FOR THE MODEL IN (A) (NOT AVAILABLE ON IBM MACHINE)

Assume file ABC.DAT contains the above model.
Run LINKFF (an independent link included in NASTRAN delivery tape, NAST01).
Answer a new file name when asked, such as DEF.DAT.
Enter 'READFILE ABC.DAT'.
A fixed-field deck will be generated and saved in DEF.DAT (approx. 15020 cards).

(C) SUMMARY OF NASTRAN TIMINGS USING ABOVE CANTILEVER MODEL IN (A)

```
VAX TIMING, COSMIC VAX/780 VMS 4.7 -
(Runs were made before November 1988)

VAX NASTRAN 87 RELEASE TIMING:
17:21:49      3.0 ELAPSED SEC       1.0 CPU SEC   TTIO     | TIMING COMPUTATION
17:22:32     46.0 ELAPSED SEC      26.0 CPU SEC   TTLP     |
17:22:32     46.0 ELAPSED SEC      26.0 CPU SEC   XCSA     | 32.0-  26.= 6.0 SEC
17:22:42     56.0 ELAPSED SEC      32.0 CPU SEC   IFP1     | 34.0-  32.= 2.0 SEC
17:22:46     60.0 ELAPSED SEC      34.0 CPU SEC   XSOR     | 7495.-  34.=7461 SEC
19:36:17   8071.0 ELAPSED SEC    7495.0 CPU SEC   IFP  BEGN
20:04:48   9782.0 ELAPSED SEC    9190.0 CPU SEC   IFP  END| 9190.-7495.=1695 SEC
20:04:48   9782.0 ELAPSED SEC    9190.0 CPU SEC   XGPI     |

VAX NASTRAN 88 RELEASE (WITH DIAG 42 ON, USING OLD XSORT) TIMING:
14:28:00      2.0 ELAPSED SEC       1.0 CPU SEC   TTIO     |
14:28:00      2.0 ELAPSED SEC       1.0 CPU SEC   XCSA     | 7.0-   1.0= 6.0 SEC
14:28:10     12.0 ELAPSED SEC       7.0 CPU SEC   IFP1     | 9.0-   7.0= 2.0 SEC
14:28:14     16.0 ELAPSED SEC       9.0 CPU SEC   XSOR     | 8334.-  9.0=8325 SEC
17:00:16   9138.0 ELAPSED SEC    8334.0 CPU SEC   IFP  BEGN
17:32:40  11082.0 ELAPSED SEC    9893.0 CPU SEC   IFP  END| 9893.-8334.=1559 SEC
17:32:40  11082.0 ELAPSED SEC    9893.0 CPU SEC   XGP      |

VAX NASTRAN 88 RELEASE TIMING:
14:13:05      2.0 ELAPSED SEC       0.0 CPU SEC   TTIO     |
14:13:05      2.0 ELAPSED SEC       0.0 CPU SEC   XCSA     |   6.0- 0.0=  6.0 SEC
14:13:14     11.0 ELAPSED SEC       6.0 CPU SEC   IFP1     |   8.0- 6.0=  2.0 SEC
14:13:18     15.0 ELAPSED SEC       8.0 CPU SEC   XSOR     | 160.0- 8.0=154.0 SEC
14:16:50    227.0 ELAPSED SEC     160.0 CPU SEC   IFP  BEGN
14:20:09    426.0 ELAPSED SEC     337.0 CPU SEC   IFP  END| 337.0-160.=177.0 SEC
14:20:09    426.0 ELAPSED SEC     337.0 CPU SEC   XGPI     |

VAX NASTRAN 88.5 NEW VERSION TIMING:
13:43:36     40.0 ELAPSED SEC      26.0 CPU SEC   TTLP     |
13:43:37     41.0 ELAPSED SEC      26.0 CPU SEC   XCSA     |  32.0-26.0=  4.0 SEC
13:43:46     50.0 ELAPSED SEC      32.0 CPU SEC   IFP1     |  33.0-32.0=  1.0 SEC
13:43:49     53.0 ELAPSED SEC      33.0 CPU SEC   XSOR     | 149.0-33.0=116.0 SEC
13:46:52    236.0 ELAPSED SEC     149.0 CPU SEC   IFP  BEGN
13:48:42    346.0 ELAPSED SEC     237.0 CPU SEC   IFP  END| 237.0-149.= 88.0 SEC
13:48:42    346.0 ELAPSED SEC     237.0 CPU SEC   XGPI     |
```

VAX/780 TIMING SUMMARY TABLE:

```
+-----------------------------------------------------------------------------+
| VERSION   |   COSMIC 87    |   COSMIC 88    |   COSMIC 88    | COSMIC 88.5     |
|           |                |   OLD XSORT    |   NEW XSORT2   | AVAILABLE IN 89 |
|-----------|----------------|----------------|----------------|-----------------|
| MODULE    |      CPU       |      CPU       |      CPU       |       CPU       |
|-----------|----------------|----------------|----------------|-----------------|
| XCSA      |        6.0     |        6.0     |        6.0     |       4.0       |
| IFP1      |        2.0     |        2.0     |        2.0     |       1.0       |
| XSORT     |     7461.0     |     8325.0     |      154.0     |     116.0       |
| IFP       |     1695.0     |     1559.0     |      177.0     |      88.0       |
+-----------------------------------------------------------------------------+
```

VAX TIMING, COSMIC MICRO-VAX 3600 VMS 5.0-2 -
(Runs were made after December 1988, 100K HICORE)

```
VAX NASTRAN 88.5 RELEASE (WITH DIAG 42 ON, USING OLD XSORT) TIMING:
14:13:42      1.0 ELAPSED SEC      0.0 CPU SEC  TTIO     | TIMING COMPUTATION
14:13:42      1.0 ELAPSED SEC      0.0 CPU SEC  XCSA     |  2.0- 0.0=  2.0 SEC
14:13:46      5.0 ELAPSED SEC      2.0 CPU SEC  IFP1     |  2.0- 2.0=  0.0 SEC
14:13:48      7.0 ELAPSED SEC      2.0 CPU SEC  XSOR     | 1273.- 2.0=1271. SEC
14:41:49   1688.0 ELAPSED SEC   1273.0 CPU SEC  IFP  BEGN
14:43:14   1773.0 ELAPSED SEC   1321.0 CPU SEC  IFP  END| 1321.-1273.=48.0 SEC
14:43:14   1773.0 ELAPSED SEC   1321.0 CPU SEC  XGPI     |

VAX NASTRAN 88 RELEASE TIMING:
09:30:48      1.0 ELAPSED SEC      1.0 CPU SEC  TTIO     |
09:30:48      1.0 ELAPSED SEC      1.0 CPU SEC  XCSA     |  3.0- 1.0=  2.0 SEC
09:30:52      5.0 ELAPSED SEC      3.0 CPU SEC  IFP1     |  4.0- 3.0=  1.0 SEC
09:30:54      7.0 ELAPSED SEC      4.0 CPU SEC  XSOR     | 62.0- 4.0= 58.0 SEC
09:32:48    121.0 ELAPSED SEC     62.0 CPU SEC  IFP  BEGN
09:34:11    204.0 ELAPSED SEC    129.0 CPU SEC  IFP  END| 129.0-62.0= 67.0 SEC
09:34:11    204.0 ELAPSED SEC    129.0 CPU SEC  XGPI     |

VAX NASTRAN 88.5 NEW VERSION TIMING:
09:37:22      1.0 ELAPSED SEC      1.0 CPU SEC  TTIO     |
09:37:22      1.0 ELAPSED SEC      1.0 CPU SEC  XCSA     |  3.0- 1.0=  2.0 SEC
09:37:26      5.0 ELAPSED SEC      3.0 CPU SEC  IFP1     |  3.0- 3.0=  0.0 SEC
09:37:28      7.0 ELAPSED SEC      3.0 CPU SEC  XSOR     | 48.0- 3.0= 45.0 SEC
09:39:06    105.0 ELAPSED SEC     48.0 CPU SEC  IFP  BEGN
09:39:53    152.0 ELAPSED SEC     79.0 CPU SEC  IFP  END|  79.0-48.0= 31.0 SEC
09:39:53    152.0 ELAPSED SEC     79.0 CPU SEC  XGPI     |
MICRO-VAX 3600 TIMING SUMMARY TABLE:
+---------------------------------------------------------------------------+
| VERSION   |  COSMIC 88    |  COSMIC 89    |  COSMIC 88    | COSMIC 88.5     |
|           |  OLD XOSRT    |  OLD XSORT    |  NEW XSORT2   | AVAILABLE IN 89 |
|-----------|---------------|---------------|---------------|-----------------|
| MODULE    |      CPU      |      CPU      |      CPU      |       CPU       |
|-----------|---------------|---------------|---------------|-----------------|
| XCSA      |               |        2.0    |        2.0    |       2.0       |
| IFP1      |      N/A      |        0.0    |        2.0    |       0.0       |
| XSORT     |               |     1271.0    |       58.0    |      45.0       |
| IFP       |               |       48.0    |       67.0    |      31.0       |
+---------------------------------------------------------------------------+


IBM TIMINGS, MSFC IBM 3084 -

IBM NASTRAN 87 RELEASE TIMING:
*      0.3 ELAPSED    0.088 CPU     TTIO        |     TIMING COMPUTATION
*      0.4 ELAPSED    0.088 CPU     XCSA        |   0.241-0.088=  0.153 SEC
*      1.4 ELAPSED    0.241 CPU     IFP1        |   0.330-0.241=  0.092 SEC
*      2.4 ELAPSED    0.330 CPU     XSOR        |  222.975-0.330=222.645 SEC
*    274.1 ELAPSED  222.975 CPU     IFP   BEGN  |
*    407.4 ELAPSED  319.521 CPU     IFP   END   | 319.521-222.975= 96.546 SEC
*    407.4 ELAPSED  319.522 CPU     XGP         |

IBM NASTRAN 88 RELEASE TIMING (WITH DIAG 42 ON, USING OLD XSORT):
*      0.9 ELAPSED    0.093 CPU     TTIO        |
*      0.9 ELAPSED    0.094 CPU     XCSA        |   0.227-0.094=  0.133 SEC
*      2.2 ELAPSED    0.227 CPU     IFP1        |   0.306-0.227=  0.079 SEC
*      3.2 ELAPSED    0.306 CPU     XSOR        |  222.801-0.306=222.495 SEC
*    252.6 ELAPSED  222.801 CPU     IFP   BEGN  |
*    339.5 ELAPSED  297.420 CPU     IFP   END   | 297.420-222.801= 74.619 SEC
*    339.5 ELAPSED  297.421 CPU     XGP         |

IBM NASTRAN 88 RELEASE TIMING (FIRST RUN):
*      0.5 ELAPSED    0.097 CPU     TTIO        |
*      0.5 ELAPSED    0.097 CPU     XCSA        |   0.215-0.097=  0.118 SEC
*      2.0 ELAPSED    0.215 CPU     IFP1        |   0.274-0.215=  0.059 SEC
*      3.6 ELAPSED    0.274 CPU     XSOR        |   8.766-0.274=  8.492 SEC
*     36.9 ELAPSED    8.766 CPU     IFP   BEGN  |
*     58.6 ELAPSED   16.813 CPU     IFP   END   |   16.813-8.766=  8.047 SEC
*     58.6 ELAPSED   16.813 CPU     XGP         |

IBM NASTRAN 88 RELEASE TIMING (SECOND RUN):
*      0.9 ELAPSED    0.100 CPU     TTIO        |
*      0.9 ELAPSED    0.100 CPU     XCSA        |   0.221-0.100=  0.121 SEC
*      2.3 ELAPSED    0.221 CPU     IFP1        |   0.282-0.221=  0.061 SEC
*      3.3 ELAPSED    0.282 CPU     XSOR        |   8.819-0.282=  8.537 SEC
*     31.9 ELAPSED    8.819 CPU     IFP   BEGN  |
*     51.9 ELAPSED   16.984 CPU     IFP   END   |   16.984-8.819=  8.165 SEC
*     51.9 ELAPSED   16.984 CPU     XGPI        |

IBM NASTRAN 88.5 RELEASE TIMING:
*      1.4 ELAPSED    0.118 CPU     TTIO        |
*     10.2 ELAPSED    0.540 CPU     TTLP        |
*      10.2 ELAPSED     0.580 CPU      XCSA         |     0.703-0.580=  0.123 SEC
*      12.9 ELAPSED     0.703 CPU      IFP1         |     0.742-0.703=  0.039 SEC
*      14.4 ELAPSED     0.742 CPU      XSOR         |     8.135-0.742=  7.393 SEC
*      75.0 ELAPSED     8.135 CPU      IFP    BEGN  |
*     139.0 ELAPSED    14.040 CPU      IFP    END   |    14.040-8.135=  5.905 SEC
*     139.0 ELAPSED    14.040 CPU      XGPI         |
```

IBM TIMING SUMMARY TABLE:

VERSION	COSMIC 87	COSMIC 88 OLD XSORT	COSMIC 88 NEW XSORT2	COSMIC 88.5 AVAILABLE IN 89
MODULE	CPU	CPU	CPU*	CPU
XCSA	0.153	0.133	0.120	0.123
IFP1	0.092	0.079	0.060	0.039
XSORT	222.645	222.495	8.515	7.393
IFP	96.546	74.619	8.106	5.905

* AVERAGE OF TWO RUNS.

UNIVAC TIMING TEST NOT AVAILABLE

CRAY TIMINGS, MODEL X-MP (COS), COURTESY OF RPK CORPORATION -
(Log files slightly edited)

CRAY NASTRAN 88 RELEASE TIMING (WITH DIAG 42 ON, USING OLD XSORT):

```
WALL CLOCK        TOTAL        INCREMENTAL      MODULE
   TIME        CPU SECONDS    CPU SECONDS      & STATUS
-------------------------------------------------------
 10:17:29         0.018          0.000         TTIO      |   TIMING COMPUTATION
 10:17:31         0.349          0.331         TTLP      |
 10:17:31         0.439          0.090         XCSA      |   0.618-  0.439=  0.179
 ID    TEN THOUSAND GRID POINTS TEST PROBLEM, OPEN CORE? (CRAY)
 10:17:34         0.618          0.179         IFP1      |   0.644-  0.618=  0.026
 TITLE = DATA INTENTIONALLY GENERATED NOT IN SORTED ORDER
 10:17:34         0.644          0.027         XSOR      | 108.788-  0.644=108.144
 10:22:24       108.788        108.144         IFP  BEGN
 10:23:43       141.799         33.010         IFP  END  | 141.799-108.788= 33.011
 10:23:43       141.799          0.000         XGPI      |
```

CRAY NASTRAN 88 RELEASE TIMING:

```
WALL CLOCK        TOTAL        INCREMENTAL      MODULE
   TIME        CPU SECONDS    CPU SECONDS      & STATUS
-------------------------------------------------------
 10:05:16         0.018          0.000         TTIO      |
 10:05:19         0.346          0.328         TTLP      |
10:05:19          0.435          0.090       XCSA     |  0.609-0.435=0.174 SEC
ID    TEN THOUSAND GRID POINTS TEST PROBLEM, OPEN CORE? (CRAY)
10:05:23          0.609          0.174       IFP1     |  0.626-0.609=0.017 SEC
TITLE = DATA INTENTIONALLY GENERATED NOT IN SORTED ORDER
10:05:23          0.626          0.017       XSOR     |  5.539-0.626=4.913 SEC
10:05:41          5.539          4.913       IFP  BEGN
10:05:57         10.575          5.035       IFP  END | 10.575-5.539=5.036 SEC
10:05:57         10.575          0.000       XGPI     |
```

```
CRAY TIMING SUMMARY TABLE:
+--------------------------------------------------------------------------+
| VERSION  |  COSMIC 87   |  COSMIC 88   |  COSMIC 88   |  COSMIC 88.5     |
|          |              |  OLD XSORT   |  NEW XSORT2  | AVAILABLE IN 89  |
|----------|--------------|--------------|--------------|------------------|
| MODULE   |     CPU      |     CPU      |     CPU      |       CPU        |
|----------|--------------|--------------|--------------|------------------|
| XCSA     |              |    0.179     |    0.174     |                  |
| IFP1     |     N/A      |    0.026     |    0.017     |       N/A        |
| XSORT    |              |  108.144     |    4.913     |                  |
| IFP      |              |   33.011     |    5.036     |                  |
+--------------------------------------------------------------------------+
```

```
CDC TIMINGS, CYBER 855 AT LRC, COURTESY OF JOE WALZ -

CDC NASTRAN 88 RELEASE TIMING WITH DIAG 42 ON (OLD XSORT):

WALL        ELAPSED     CPU
CLOCK       SECONDS   SECONDS  MODULE
=====================================
18.50.41.      3.0      1.245  TTIO      |    TIMING COMPUTATION
18.50.57.     19.0      5.698  TTLP      |
18.50.58.     20.0      6.289  XCSA      |    10.532-  6.289=   4.243 SEC
18.51.05.     27.0     10.532  IFP1      |    11.876- 10.532=   1.344 SEC
18.51.07.     29.0     11.876  XSOR      |   650.388- 11.876= 638.512 SEC
19.16.03.   1525.0    650.388  IFP  BEG  |
20.27.41.   5823.0   3083.137  IFP  END  |  3083.137-650.388=2432.749 SEC
20.27.41.   5823.0   3083.141  XPGI      |

CDC NASTRAN 88 RELEASE TIMING (NEW XSORT2):
08.26.25.      3.0      1.278  TTIO      |
08.26.40.     18.0      5.895  TTLP      |
08.26.41.     19.0      6.501  XCSA      |     9.730-  6.501=   3.229 SEC
08.26.45.     23.0      9.730  IFP1      |    10.422-  9.730=   0.692 SEC
08.26.47.     25.0     10.422  XSOR      |   695.797- 10.422= 685.375 SEC
08.45.31.   1149.0    695.797  IFP  BEGN |
09.01.24.   2102.0   1335.940  IFP  END  |  1335.940-695.797= 640.143 SEC
09.01.24.   2102.0   1335.944  XGPI      |

CDC NASTRAN 88.5 RELEASE TIMING (TEST RUN BY G.CHAN):
03.10.50.      1.0       .184  TTIO      |
03.10.55.      6.0      4.541  TTLP      |
03.10.56.      7.0      5.127  XCSA      |     8.341-  6.501=   1.840 SEC
```

```
03.10.58.    9.0      8.341  IFP1      |     8.615-  8.341=   0.274 SEC
03.10.59.   10.0      8.615  XSOR      |   408.704-  8.615= 400.029 SEC
03.15.55.  306.0    408.704  IFP  BEGN |
03.19.24.  515.0    708.021  IFP  END  |   708.021-408.704= 299.317 SEC
03.19.24.  515.0    708.025  XGPI      |
```

CDC TIMING SUMMARY TABLE:

VERSION	COSMIC 88 OLD XSORT		COSMIC 88 NEW XSORT2		COSMIC 88.5 NEW XSORT2	
MODULE	ELAPS	CPU	ELAPS	CPU	ELAPS	CPU
XCSA	7	4.2	4	3.2	1	1.84
FP1	2	1.3	2	0.7	2	0.27
XSORT	1493	638.5	1124	685.4	296	400.03
IFP	4298	2432.7	953	640.1	209	299.32
TOT.CPU	3141.4		1337.8		713.9	
*SRU's	4259		1839		976	

* CDC COMPUTER CHARGE IS BASED ON SRU UNITS.

APPENDIX B

USERS' MANUAL UPDATE PAGES FOR INPUTT5 MODULE

VI. PARAMETERS:

1. The meanings of the first three parameter values (P1, P2, P3) are the same as those described for INPUTT2 Module, except (1) values -5 through -8 for P1 are not available, and a new P1=-9 to rewind input tape; and (2) the user file code and the FORTRAN file name are given below. (The default value for P2 is 16, or 12 for a CDC computer.)

FORTRAN LOGICAL UNIT, P2	USER FILE CODE
11	UT1 (CDC only)
12	UT2 (CDC only)
14	INPT (UNIVAC,VAX)
15	INP1 (All
16	INP2 machines
:	: except
23	INP9 CDC)
24	INPT (IBM only)

2. The fourth parameter (P4) for this module is used to specify whether the user tape was written with formats (P4=1), or binary tape (P4=0). Default is P4=0.

VII. METHODS:

Since INPUTT5 is intended to be a companion module to OUTPUT5, it is therefore suggested that the user should refer to the Methods and Remarks sections of the OUTPUT5 module for input tape structure.

Subroutine INPTT5 is the main driver for the INPUTT5 module. Its primary function is to read matrix data blocks from the user input tape. When a table data block is encountered, INPTT5 calls subroutine TABLEV to process the data. The user input tape always begins with a tape ID record which tells when the tape was generated, on what machine, tape identification, formatted or unformatted tape, and NASTRAN system buffer size. This tape ID record can be skipped, or read by the following FORTRAN code:

```
       INTEGER TAPEID(2),MACHIN(2),DATE(3),BUFSIZ,P4X
       READ (TAPE   ) TAPEID,MACHIN,DATE,BUFSIZ,P4X    or
       READ (TAPE,10) TAPEID,MACHIN,DATE,BUFSIZ,P4X
    10 FORMAT (2A4,2A4,3I8,I8,I8)
```

I. NAME: INPUTT5 (Reads A User-Written FORTRAN File, Formatted or Unformatted)
(The companion module is OUTPUT5)

II. PURPOSE: Recovers up to five data blocks from a FORTRAN-written user file, formatted or unformatted. (The FORTRAN file may reside either on physical tape or on a mass storage device.) This file may be written either by a user-written FORTRAN program or by the companion module OUTPUT5. The Programmers' Manual describes the format of the user tape which must be written in order to be readable by INPUTT5. The unformatted binary tape can only be read by a computer of the same manufacturer as the one that created the tape. The formatted tape can be created and read by different computers (CDC, UNIVAC, IBM, and VAX). The data blocks to be recovered can be matrices, tables, or both.

III. DMAP CALLING SEQUENCE:

```
INPUTT5  /DB1,DB2,DB3,DB4,DB5/C,N,P1/C,N,P2/C,N,P3/C,N,P4 $
```

INPUTT5 is intended to have the same logical action as the FORTRAN User File module INPUTT2 and the GINO User File module INPUTT1 except for formatted tape. It is therefore suggested that the examples shown under modules INPUTT2 and OUTPUT1 be used for OUTPUT5 as well, excepting the addition of the P4 parameter.

IV. INPUT DATA BLOCKS:

Input data blocks are not used in this module call statement.

V. OUTPUT DATA BLOCKS:

DBi are data blocks which will be recovered from one of the NASTRAN tape files INP1, INP2 through INP9 (UT1, UT2 for CDC computer). Any or all of the output data blocks may be purged. Only non-purged data blocks will be taken from the user tape. The data blocks will be taken sequentially from the tape starting from a position determined by the value of the first parameter. Note that any purged output file will cause skipping of a corresponding file in the user input tape. The output data block sequence A,B,,, is not equivalent to ,A,,B, or ,,,A,B.

UNFORMATTED TAPE:

The rest of the unformatted tape can be read by the following FORTRAN code:

```
READ (TAPE) L,J,K,(ARRAY(I),I=J,K)
```

where L is a control word;
L = 0, ARRAY contains matrix (or table) header record
=+n, ARRAY contains data for the nth column of the matrix
=-1, ARRAY contains end of matrix record.
The ARRAY below J and above K are zeros.

The matrix header record and the table header record (L=0) differ only on the 5th and 6th words of ARRAY. If both words are zeros, it is a table header, and the entire table data can be read by:

```
READ (TAPE) L,(ARRAY(I),I=1,L)
```

where ARRAY may contain integers, BCD words, and real single and double precision numbers.

Table data ends with a (1,0.0) record.

FORMATTED TAPE:

For matrix data, the rest of the formatted tape can be read by:

```
   READ (TAPE,20) L,J,K,(ARRAY(I),I=J,K)
20 FORMAT (3I8,/,(10E13.6))     (for single precision data) or
20 FORMAT (3I8,/,(5D26.17))     (for double precision data)
```

where the control words L, J, and K are the same as in the unformatted case, and th data type, single or double precision, is determined already by the 4th word of the matrix trailer embedded in the matrix header record. (See Remark 5 of OUTPUT5 module)

For table data, the rest of the formatted tape can be read by:

```
   CHARACTER*5 ARRAY(500)
   READ (TAPE,30) J,(ARRAY(I),I=1,J)
30 FORMAT (I10,24A5,/,(26A5))
```

Notice the formatted record was written in the units of 5-byte character words, and the first byte of each unit indicates what data type follows. The following table summarizes the method to decode the character data in ARRAY.

FIRST BYTE	DATA TYPE OF ARRAY	UNITS USED	FORMAT
'/'	BCD word	1	A4
'I'	Integer	2	I9
'R'	Real, s.p.	3	E14.7
'D'	Real, d.p.	3	D14.7
'X'	Filler	1	4X

Table data ends with a (1,'0') record.

VIII. EXAMPLES:

```
$ COPY KJI AND KGG TO INP1 (UNIT 15), SEQUENTIAL FORMATTED TAPE
  OUTPUT5 KJI,KGG,,,//-1/15/*MYTAPE*/1 $
$ RECOVER THE 2 FILES FROM INP1 (UNIT 15) AND MAKE THEM NASTRAN GINO FILES
  INPUTT5 /OKJI,OKGG,,,/-1/15/*MYTAPE*/1 $
```

VIII. REMARKS:

1. Since open core is used to receive data from user input tape, INPUTT5 can handle all kinds and all sizes of data blocks.

2. The UNIVAC and VAX users should read the Important Note at the end of the description of the INPUTT2 module.

APPENDIX C

USERS' MANUAL UPDATE PAGES FOR OUTPUT5 MODULE

I. NAME: OUTPUT5 (Creates A User-Written FORTRAN File, Formatted or Unformatted)
(The companion module is INPUTT5)

II. PURPOSE: Writes up to five NASTRAN GINO data blocks to a user FORTRAN file using a FORTRAN write, formatted or unformatted. (The FORTRAN file may reside either on physical tape or on a mass storage device.) If the data block contains matrix data, each matrix column is first unpacked, then written out to the user file in unpacked form. If the data block contains table data and formatted records are requested, a dynamic scheme is used to generate the appropriate format for the FORTRAN write. Coded symbols are also included in the formatted table data, so that they can be read back into the NASTRAN system by the INPUTT5 module, or by a user written FORTRAN program. Mixed matrix and table data blocks are allowed in one OUTPUT5 operation.

The unformatted (binary) user file is intended to be used later in the same computer, or a similar computer of the same manufacturer. The formatted file can be generated in one computer system and used later in another, with complete freedom in operating systems and computer manufacturers. The formatted file can be viewed and edited by the use of the system editor. The records contain 132 characters (or less) per line.

The parameters in OUTPUT5 are modeled after OUTPUT2. They can be used to direct which user output file (INP1, INP2, UT1 etc.) is to be used, to write formatted or unformatted records, to position the output file prior to writing, and to place an End-Of-File mark at the end of the tape. Multiple calls are allowed. The user is cautioned to be careful when positioning the user output file with OUTPUT5, since he may inadvertently destroy information through improper positioning. Even though no data blocks are written, an EOF will be written at the completion of each call, which has the effect of destroying anything on the tape forward of the current position.

III. DMAP CALLING SEQUENCE:

```
OUTPUT5  DB1,DB2,DB3,DB4,DB5//C,N,P1/C,N,P2/C,N,P3/C,N,P4/C,N,T1/C,N,T2/
         C,N,T3/...C,N,T10 $
```

OUTPUT5 is intended to have the same logical action as the FORTRAN User File module OUTPUT2 and the GINO User File module OUTPUT1, except for formatted tape. It is therefore suggested that the examples shown under modules OUTPUT2 and OUTPUT1 be used for OUTPUT5 as well, excepting the addition of the P4 parameter. All samples should be ended with a call to OUTPUT5 with P1=-9.

IV. INPUT DATA BLOCKS:

DBi - Any data block which the user desires to be written on one of the NASTRAN FORTRAN user files INPT, INP1, INP2,..., INP9. Any or all of the input data blocks may be purged. Only unpurged data blocks will be placed on the user file.

V. OUTPUT DATA BLOCKS: None.

VI. PARAMETERS:

1. The meanings of the first three parameter values (P1, P2, P3) are the same as those described for the OUTPUT2 Module, except the user file code and the FORTRAN file name are given below. (The default value for P2 is 15, or 11 for a CDC machine.)

FORTRAN LOGICAL UNIT, P2	USER FILE CODE
11	UT1 (CDC only)
12	UT2 (CDC only)
14	INPT (UNIVAC,VAX)
15	INP1 (All
16	INP2 machines
:	: except
23	INP9 CDC)
24	INPT (IBM only)

2. The fourth parameter (P4) for this module is used to specify whether the user output tape is to be written formatted (P4=1), or unformatted (P4=0, default). Unless the tape is to be used later by a different computer or a different operating system, the unformatted tape should be used.

3. The 10 Ti parameters (T1, T2, T3,..., T10) are used only for table data blocks. They are used only when a formatted output file is requested (P4=1), and the user wants to override the automatic format generation of the OUTPUT5 module. (Default - all Ti are zeros)

   The following rules are used to create user-directed output format:

   a. 9 digits must be specified on a Ti parameter. Zero fill if necessary.
   b. The digits are continued among the Ti parameters; therefore up to 90 digits are allowed. The digits are arranged from left to right. First digit specifies the

format of the first data word. Second, third, fourth, etc., specify the second, third, fourth data words, etc. (See exception below using digits 5 through 9)

c. The values of digits and their meanings are -

   0, format not specified; whatever format OUTPUT5 generated will be used,
   1, specifies integer format,
   2, specifies single precision real format,
   3, specifies BCD format,
   4, specifies double precision real format, and
   5-9, specify multiple format of the same type indicated by next digit, which must be 0 through 4.
   e.g. 061352000 is same as 0111111322222000

VII. METHODS:

The methods used to transfer data from NASTRAN GINO data blocks to the user output tape (or file) depend on whether

a. the data blocks are matrix or table,
b. formatted or unformatted output tape is requested, and
c. data contains single precision real numbers or double precision numbers, or both. (Table data block only)

The methods used must also guarantee continuity of mixed matrix and table types of block data on the user output tape. That is, the mixed data must be able to be read back into the NASTRAN system, or processed by a user's program, by a common switching mechanism.

OUTPUT5 treats any input data block as matrix if the 5th and the 6th words (maximum non-zero matrix column length and matrix density) are both non-zero. Otherwise, the data block is table. This method is, however, not perfect. Most table data blocks generated by LINK1, such as GEOM1, GEOM2, EPT, MPT, etc. may have non-zero 5th and 6th trailer words.

UNFORMATTED TAPE -

The data transfer from a GINO file to an unformatted tape is comparatively simple. The difference in processing matrix data and table data lies in a single key word of the length of each record.

MATRIX - A matrix header record that includes the original GINO trailer is written to user tape first. Thus the total number of records (equal number of columns) and the length of each record (equal number of rows) are known. Each column of the matrix is unpacked and copied out to the user tape, except that the leading and trailing zeros are not copied out. The data is either single precision or double precision real numbers. Each output record is also preceded by three control words. The following FORTRAN code can read one such column array (the ICOL matrix column):

```
READ (TAPE) ICOL,JB,JE,(ARRAY(J),JB,JE)
```

TABLE - A table header record, with the 5th and 6th trailer words set to zeros, is also written out to indicate the following records are of table type. Records from the input GINO data block are read and transferred to user tape directly, except each output record is preceded by one additional word, which tells the total length of this current record. The following FORTRAN code can be used to read one such record:

```
READ (TAPE) LENGTH,(ARRAY(J),J=1,LENGTH)
```

FORMATTED TAPE -

Most of the attributes of unformatted tape apply equally well to the formatted tape, except tapes are written with FORTRAN formats.

MATRIX - All integers are written in I8 format, BCD in A4 format, single precision real numbers in E13.6, and double precision numbers in D26.17. Only the matrix header record can have all four data types; the matrix column records contain only real numbers. The following FORTRAN code reads the header record and/or a matrix column:

```
    WRITE (TAPE,10) I,J,K,(A(L),L=J,K)
10  FORMAT (3I8,/,(10E13.6 ))     (for single precision data) or
10  FORMAT (3I8,/,( 5D26.17))     (for double precision data)
```

TABLE - All integers are written in ('I',I9) format, BCD in ('/',A4) format, single precision real numbers in ('R',E14.7) format, and double precision numbers in ('D',E14.7). Notice that 5 bytes are used for BCD, 10 bytes for integer, and 15 bytes for real numbers, single or double precision. NASTRAN table data blocks often contain integers, BCD, and single and double precision real numbers in a mixed fashion. Each table record may have a different table length. To write formatted NASTRAN tables and to read them back later present a real challenge in FORTRAN programming. The OUTPUT5 module calls subroutine TABLE5 to process table data, and the INPUTT5 module calls subroutine TABLEV to read them back.

TABLE5 generates dynamically a unit of format - ('I',I9), ('/',A4), etc. - to match each data type - integer, BCD, etc. When the synthesized format reaches 130 characters (or bytes), a line of data is written out. A table therefore may require multiple lines (each line physically is a record). In addition, the first word of the first line contains the total length of this table. The following FORTRAN code can be used to read back a table from the user tape into 5-character ARRAY:

```
      CHARACTER*5 ARRAY(500)
      READ (TAPE,20) LENGTH,(ARRAY(J),J=1,LENGTH)
   20 FORMAT (I10,24A5,/,(26A5))
```

The first byte of each 5-character ARRAY (which is I,/,R, or D) can be used to convert the 5-, 10-, or 15-character data back to BCD, integer, or real numbers (single or double precision). For more details, see INPUTT5 module and INPTT5 FORTRAN source subroutine.

TABLE5 calls subroutine NUMTYP to determine the data type, then issue the corresponding format for output. NUMTYP, however, is not one hundred percent foolproof. One in five or ten thousand times, NUMTYP may err in determining exactly the data type. Also, when TABLE5 passes a computer word to NUMTYP with no other information, NUMTYP cannot tell if it is part of a double precision word, or if it is a single precision word. (In this case, single precision word is assumed.) Finally, NUMTYP cannot distinguish between integer zero and real number zero. (A period may be important in the output format). TABLE5 therefore may generate the wrong format due to NUMTYP's internal limitations.

In case that TABLE5 does produce erroneous format, the user can override the automatic format generation by the Ti parameters which supply OUTPUT5 the exact format to use, in a condensed, coded form. 90 (or more if 5, 6, 7, 8, or 9 are used in the Ti specification) unit formats can be specified.

The following example illustrates the use of the Ti parameter.

```
Data on table:
3  4  3.4  5.0E-3  TESTING  .6D+7  9  G  3.2  8  0.  0  4 12 13  14  15  28  61   88
14   44 .7D+7

Ti specification:
T1=112233413, T2=212516140  or
T1=604000025, T2=060400000 (7th and 24th words are d.p.
                            and 12th word is real)
NOTE - 2 BCD words in 'TESTING',
       all others are 1 computer word per data entry.
       T2, the last Ti used here, must fill up with zeros to make up a 9-digit word.
```

When viewed with a system editor, the above example looks like this (first line):

```
37I        3I         4R 5.0000000E-3/TEST/ING D 6.0000000D+07 etc.
++----------++++++++++----------------++++++++++-----------------
     1st       2nd          3rd           4th            5th data etc.
```

The first 37 indicates there are 37 5-byte words in this record.
the '++----' line and the '1st,2nd...' line are added here for video purposes.

Since the formatted data line may not end exactly at 130 bytes, one or two fillers of the form 'X' and four blanks may appear at the end of an output line.

The matrix data blocks are handled by the main routine OUTPT5. OUTPT5 calls TABLE5 only when the former encounters a table data block input.

VIII. EXAMPLES:

```
$ Copy KJI, KGG, and CASECC to INP2 (unit 16), sequential formatted tape
  OUTPUT5 KJI,KGG,CASECC,,//-1/16/*MYTAPE*/1  $
$ Recover the files from INP2 (unit 16) and make them NASTRAN GINO files
  INPUTT5 /OKJI,OKGG,OCASECC,,/-1/16/*MYTAPE*/1  $
```

IX. REMARKS:

1. Formatted tape (P4=1) takes a longer time and more space to write than the unformatted tape. Unless the tape is intended to be used later by a different computer, unformatted tape should be selected (P4=0).

2. The OUTPUT5 'records' are written to tape 'identically' with both formatted and unformatted FORTRAN write commands. The matrix header and the table header can be read 'identically' without prior knowledge of what type of data, matrix or table, is coming up next.

3. All matrix records are written to tape in a standard way, except the first matrix header record.

   All table records are written to tape in a standard way, including table header record and the last ending record.

4. The first tape header record is composed of 9 words as shown below:

RECORD	WORD	CONTENTS	P4=0	P4=1
0	1,2	Tapeid (=P2)	2*BCD	2A4
	3,4	Machine (CDC,UNIVAC,IBM,VAX)	2*BCD	2A4
	5-7	Date	3*INT	3I8
	8	System BUFFER SIZE	INT	I8
	9	P4 used in creating tape (0,1)	INT	I8

5. This remark and the next one deal only with matrix data blocks.

Three types of data records follow the header record, or the EOF record of a previous data block. They are:

a. Matrix header record
b. Matrix column data record
c. EOF record

These records are written to tape in a standard procedure. Three control words are written out first, followed by the actual data. Binary FORTRAN write is used in unformatted tape (P4=0), and each logical record holds a complete set of data. The following FORTRAN statement is used to write the entire data record:

```
      WRITE (TAPE) I,J,K,(A(L),L=J,K)
```

For formatted tape, multiple logical records are actually written for each complete set of data. The following FORTRAN statements are used to write the entire data record:

```
      WRITE (TAPE,30) I,J,K,(A(L),L=J,K)
   30 FORMAT (3I8,/,(10E13.6))      (for single precision data) or
   30 FORMAT (3I8,/,(5D26.17))      (for double precision data)
```

In the above WRITE statements, the value of I is used to indicate the type of record just read.

VALUE OF I	TYPE OF RECORD
0	Matrix header record
+n	Nth matrix column data
-1	End-of-matrix

The column data is written to tape from the first non-zero row position (J) to the last non-zero row position (K). The following table describes the contents of the data records written to tape by the OUTPUT5 module.

RECORD+	WORD	CONTENTS	P4=0	P4=1
1		Matrix header record -		
	1	0	INT	I8
	2,3	1,1	2*INT	2I8
	4	0.0	F.P.	E13.6 or D26.17
	5-10	Matrix trailer (Col,Row,Form,Type,Max,Density)	6*INT	6I8
	11,12	DMAP Name of DB1	2*BCD	2A4
2	1	1 (First matrix column)	INT	I8
	2	Row pos. of first non-zero elem.	INT	I8
	3	Row pos. of last non-zero elem.	INT	I8
	4-W	First banded column data (W=Word3-Word2)	6*INT	(**)
3	1	2 (Second matrix column)	INT	I8
	2	Row pos. of first non-zero elem.	INT	I8
	3	Row pos. of last non-zero elem.	INT	I8
	4-W	Second banded column data	6*INT	(**)
4	1	3 (Third matrix column)	INT	I8
	2	Row pos. of first non-zero elem.	INT	I8
	3	Row pos. of last non-zero elem.	INT	I8
	4-W	Third banded column data	6*INT	(**)
:	:	:		
L	1	L-1 (last matrix column)	INT	I8
	2	Row pos. of first non-zero elem.	INT	I8
	3	Row pos. of last non-zero elem.	INT	I8
	4-W	Last banded column data	6*INT	(**)
L+1	1	-1	INT	I8
	2,3	1,1	2*INT	2I8
	4	0.0	F.P.	D26.17

(Repeat records 1 through L+1 for next matrix data block.)

Where (**) is (10E13.6) or (5D26.17).

(+RECORD No. does not correspond one to one to the actual physical record No.)

6. A record of (n,1,1,0.0) is written out for a null Nth column.

7. This remark deals only with table data blocks.

   Three types of data record follow the header record, or an EOF record of previous data block. They are:

   a. Table header record
   b. Record(s) of a table (a table data block can have more than one table record)
   c. EOF record.

   The table header record has a general structure as in the standard procedure for the matrix records, except that the 5th and 6th words of the matrix trailer section are zeros.

   The table record was discussed in great detail in the METHOD section for both formatted and unformatted output tape. A table record is created for each table in the input data block, and no skipping forward or backward is allowed on the input file.

   If double precision data are encountered in a table record, the double precision data will be truncated to single precision, but the format of ('D',E14.7) will be used. (INPUTT5 will re-generate the data back to their double precision status.)

   An End-Of-File record in the form of '-1 1 1 0.0D+0' ends the table record output.

8. Since the formatted tape (P4=1) is intended to be used in different computers, the OUTPUT5 module appends no system control word(s) to the FORTRAN written formatted records. The output tape must be unlabeled, fixed block size with record size of 132 characters, and ANSI unpacked character data set. The specification of the tape is either internally specified (UNIVAC) by a FORTRAN open statement, or uses system default tape specification (IBM and VAX). The CDC user must specify the output tape externally by the appropriate FILE, LABEL, or REQUEST cards:

   For example:

   ```
   LABEL,TAPE,NT,D=1200,CV=AS,F=S,LB=KU,PO=W.
   FILE,TAPE,MRL=132,MBL=132,RT=F,BT=C.
   ```

9. Since open core is used in data processing, the OUTPUT5 module is capable of handling all kinds and all sizes of input data blocks.

APPENDIX D

USERS' MANUAL UPDATE PAGES FOR PARAMD MODULE

DIRECT MATRIX ABSTRACTION

I. NAME: PARAMD (Parameter Processor - Double Precision)

II. PURPOSE: To perform specified arithmetic, logical, and conversion operations on double precision real or double precision complex parameters.

III. DMAP CALLING SEQUENCE:

```
PARAMD  // C,N,OP / V,N,OUTD / V,N,IND1 / V,N,IND2 / V,N,OUTC / V,N,INC1 / V,N,INC2 /
                 V,N,FLAG $
```

IV. INPUT DATA BLOCKS: None.

V. OUTPUT DATA BLOCKS:

None.

VI. PARAMETERS:

OP - Input-BCD operation code from the table below - no default
OUTD - Output D.P.-default = 0.0D+0
IND1 - Input-D.P.-default = 0.0D+0
IND2 - Input-D.P.-default = 0.0D+0
OUTC - Output-D.P.-complex-default = (0.0D+0, 0.0D+0)
INC1 - Input -D.P.-complex-default = (0.0D+0, 0.0D+0)
INC2 - Input -D.P.-complex-default = (0.0D+0, 0.0D+0)
FLAG - Output/output-integer-default= 0 (See Remark 6)

The values of parameters are dependent upon OP as shown in the table described in PARAMR module (pages 5.5-40 and 41). In addition, a new OP operation code is added:

OP	OUTPUTS
ERR	If Flag is set to 0 (or by default), NASTRAN system NOGO flag (the 3rd word of /SYSTEM/) is set to integer zero unconditionally. If FLAG is set to non-zero by user, NASTRAN job will terminate if any preceding PARAMD (or PARAMR) has non-fatal error(s).

VII. REMARKS:

1. All parameters, except OP, must be "V" type. Default parameter values will be used in case of error. Error in input parameter(s) would cause output parameter(s) to pick up the original default value(s).

2. All input errors are non-fatal, with error messages printed.

3. PARAMD does its own SAVE; therefore, a SAVE is not needed following the module.

4. For OP = DIV or OP = DIVC, the output is zero if the denominator is zero, and FLAG is set to +1.

5. For OP = SIN, OP = COS or OP = TAN, the input must be expressed in radians.

6. The default value of FLAG is zero as stated in the Programmer's manual. All NASTRAN releases prior to 1989 actually used a +1 instead of 0. The case where FLAG = -1, was not affected.

7. Remarks 1, 2, and 6 also apply to the PARAMR module. The new ERR operation code is also available in PARAMR.

VIII. EXAMPLES:

```
PARAMR  //*ERR*  $
PARAMR  //*ADD*     /V,N,R1SP4  /V,N,R1    /V,N,SP4    $
PARAMR  //*SUB*     /V,N,R1SP4  /V,N,R1    /V,N,SP4    $
PARAMR  //*ABS*     /V,N,ABSR1  /V,N,R1                $
PARAMR  //*SQRT*    /V,N,SQTR1  /V,N,ABSR1             $
PARAMR  //*MPYC* ////V,N,CMPY   /V,N,SCPLX /V,N,CS1    $
PARAMR  //*COMPLEX*//V,N,R1     /V,N,SP4   /V,N,OUTC   $
PARAMR  //*LE*     //V,N,R1     /V,N,SP4////V,N,LEFLG $
PARAMD  //*MPY*     /V,N,RDPDP  /V,N,RDPX  /V,N,RDPX   $
PARAMD  //*DIV*     /V,N,DP4X   /V,N,DP4   /V,N,RDPX   $
PARAMD  //*EXP*     /V,N,EXPX   /V,N,DP4   /V,N,RDP    $
PARAMD  //*CONJ* ////V,N,CONJX  /V,N,CDP4              $
PARAMD  //*EQ*     //V,N,EXPX   /V,N,DP4////V,N,EQFLG $
PARAMD  //*DIVC* ////V,N,DIVCX  /C,Y,DCPLX4/V,N,CDP4   $
PARAMD  //*ERR*  ////  //       /C,N,1                 $
PRTPARM // 0     $
```

APPENDIX E

USERS' MANUAL UPDATE PAGES FOR SCALAR MODULE

DIRECT MATRIX ABSTRACTION

I. <u>NAME</u>: SCALAR (Converts matrix element to parameter)

II. <u>PURPOSE</u>: To extract a specified element from a matrix for use as a parameter.

III. <u>DMAP CALLING SEQUENCE</u>:

```
SCALAR  DB // C,N,ROW/C,N,COL/V,N,RSP/V,N,RDP/V,N,CSX/V,N,CDX   $
```

IV. <u>INPUT DATA BLOCKS</u>:

DB - may be any type of matrix (single precision or double precision, real or complex)

V. <u>OUTPUT DATA BLOCKS</u>: None.

VI. <u>PARAMETERS</u>:

ROW - Row number of element to be extracted from [DB]. Integer input, default= 1

COL - Column identification of element. Integer input, default= 1

RSP - Output, value of element(ROW,COL) in single precision real, default= 0.0

RDP - Output, value of element(ROW,COL) in double precision real, default= 0.D+0

CSX - Output, value of element(ROW,COL) in single precision complex, default= (0.,0.)

CDX - Output, value of element(ROW,COL) in single precision complex, default= (0.D+0,0.D+0)

VII. <u>REMARKS</u>:

1. RSP, RDP, CSX and CDX will be set by the module whenever they are present and of the "V" type parameters. The parameters will be printed out in their respective formats according to their precision types. Warning message will be printed if type mismatch occurs or element specified is out of matrix range.

2. After execution, the parameter value will be delivered to NASTRAN's executive VPS table as a numerical value in the form specified by any of the parameters RSP, RDP, CSX, or CDX. The output parameters can also be printed by the PRTPRM module which carries normally more digits.

3. SCALAR does its own SAVE; therefore, a SAVE is not needed following the module. There is no save for any invalid parameter, and the default value remains unchanged.

4. If [DB] is purged, all parameter default values remain unchanged.

5. All the output parameters can be printed out by PRTPRM module.

6. See PARAML for similar capability.

VIII. EXAMPLES:

Obtain the value of the element in column 8 and row 2 of the matrix KLL.

```
SCALAR  KLL//C,N,2/C,N,8 /V,N,S1  $
SCALAR  KLL//C,N,2/C,N,8 //V,N,D1/V,N,S2/V,N,D2  $
```

The output parameters give the following results:

S1 = KLL(2,8), in single precision real,
D1 = KLL(2,8), in double precision real,
S2 = KLL(2,8), in single precision complex expression, and
D2 = KLL(2,8), in double precision complex expression.

APPENDIX F

USERS' MANUAL UPDATE PAGES FOR PARAML MODULE

DIRECT MATRIX ABSTRACTION

I. <u>NAME</u>: PARAML (Abstract parameters from a list)

II. <u>PURPOSE</u>: To convert an element from a GINO matrix or table data block, to a legitimate NASTRAN parameter, or parameters

III. <u>DMAP CALLING SEQUENCE</u>:

```
PARAML  DB // C,N,OP / V,N,P1 / V,N,P2 / V,N,RSP/ V,N,INT/ V,N,RDP/ V,N,BCD/ V,N,CSX/
        V,N,CDX   $
```

IV. <u>INPUT DATA BLOCKS</u>:

DB - Any GINO data block file (table or matrix, single precision or double precision, real or complex)

V. <u>OUTPUT DATA BLOCKS</u>: None

VI. <u>PARAMETERS</u>:

OP - One of the following key words, BCD input, no default. 'MATRIX', 'NULL', 'PRESENCE', 'TRAILER', 'TABLE1', 'TABLE2', or 'TABLE4'

P1,P2 - Input, see Remark 4 below, integer input, default= 1,1

P2 - Output, integer (only in OP=TRAILER)

RSP - Output, single precision real number, default= 0.0

INT - Output, integer number, default= 0

RDP - Output, double precision real number, default= 0.D+0

BCD - Output, two BCD words in 2A4 format, default= (VOID)

CSX - Output, single precision complex number, default= (0.,0.)

CDX - Output, double precision complex, default= (0.D+0,0.D+0)

VII. REMARKS:

1. RSP, INT, RDP, BCD, CSX and CDX will be set by the module whenever they are present and of the "V" type parameters. The parameters will be printed out in their respective formats according to their precision types. Warning message will be printed if type mismatch occurs or end-of-record is encountered.

2. After execution, the parameter value will be delivered to NASTRAN's executive VPS table as a numerical value in the form specified by any one or some of the parameters RSP, RDP, CSX, CDX, INT, or BCD (4 BCD characters per word, the rest of the word blank filled).

3. PARAML does its own SAVE; therefore, a SAVE is not needed following the module. Invalid parameter due to type mismatch or EOR encountered, is not saved and the default value remains.

4. P1 and P2 control the location in the data block of the element to be selected. The meaning of P1 and P2 depend on OP selection as explained in Remarks 5 through 9.

5. If OP = TABLEi (where i=1,2,OR 4), P1 is the record number and P2 is the word position of the target element in DB. Word position is based on computer word count (1 word per integer or s.p.real, 2 words per d.p.real or s.p.complex, and 4 words per d.p.complex). The table data from record P1 and word P2 (or word P2 plus more) will be delivered to the VPS table as a numerical value in the form specified.

   If OP = TABLE1, one data word from P2 word position, record P1, will be used to form the output parameter.

   If OP = TABLE2, two data words from P2 and P2+1, record P1, will be used.

   If OP = TABLE4, four words from P2, P2+1, P2+2, and P2+3, record P1, will be used.

   Since table data block DB can contain mixed types of data, the user must know ahead of time what the original data type is, and select TABLE1, TABLE2, or TABLE4 accordingly.

For example,

the data in P2, p2+1, P2+2, and P2+3 are a, b, c, d, and the output parameter request is d.p.complex CDX,
TABLE1 gives CDX = (a.D+0, 0.D+0)
TABLE2 gives CDX = (a.D+0, b.D+0)
TABLE4 gives CDX = (e.D+0, f.D+0)
where e is a d.p.real number formed by the union of a and b,
and f, by the union of c and d.

6. If OP = MATRIX, P1 is the row number and P2 is the column number of the matrix in [DB] to be read. The matrix element of (ROW,COL) will be delivered to VPS as a numerical value in the form specified by one or more of the parameters RSP, RDP, CSX, or CDX. Requests for CSX or CDX from a real matrix will assign the value of (ROW,COL) to the real part and zero to the imaginary part. The requested output parameter(s) are set to zero(s) and a warning message is issued if:
   (1) P1 and/or P2 exceed the matrix order,
   (2) requests for RSP and RDP from a complex matrix,
   (3) requests for INT and BCD from [DB],
   and the invalid output parameter(s) are not saved.

   (Notice that row first and column second is consistent with SCALAR module parameter input, and also with common practice in matrix element designation; (row,column)).

7. If OP = NULL and if [DB] is a matrix, INT is set to -1 if the sixth word of the matrix trailer, the matrix density, is zero.

8. If OP = PRESENCE, INT will be -1 if input data block is purged.

9. If OP = TRAILER, P2 is output as the value of ith word of the matrix trailer where i is set by P1 in accordance with the following table.

P1	TERM OF MATRIX TRAILER
1	Numbers of columns
2	Number of rows
3	Form of matrix
4	Precision of matrix
5	Maximum number of nonzero terms in any column of the matrix
6	Matrix density

10. One or more of the output parameters can be requested simultaneously.

11. After execution, a user information message prints out the parameter value in the format prescribed by the user. The output parameters can also be printed by the PRTPRM module which carries normally more digits. (PRTPRM may actually print integer zero in a real number format, 0.0)

12. See SCALAR module for similar capability.

VIII. EXAMPLES:

Obtain the value in column 1, row 4 of a real matrix, and record 2 word 5 of a table.

```
PARAML  KGG //*MATRIX*/C,N,4/C,N,1     /V,N,STERM  $
PARAML  KGG //*MATRIX*/C,N,4/C,N,1   ///V,N,DTERM  $
PARAML  KGG //*MATRIX*/C,N,4/C,N,1 /////V,N,CSTERM $
PARAML  KGG //*MATRIX*/C,N,4/C,N,1//////V,N,CDTERM $
PARAML  KGG //*MATRIX*/C,N,4/C,N,1/V,N,TERM1//V,N,TERM2//V,N,TERM3/V,N,TERM4 $
PARAML  CASECC //*TABLE1*/C,N,2/C,N,2  //V,N,ATERM $
PARAML  CASECC //*TABLE2*/C,N,2/C,N,5////V,N,BTERM $
```

The above output parameters yield the following results:

STERM ,TERM1 = KGG(4,1), in single precision,
DTERM ,TERM2 = KGG(4,1), in double precision,
CSTERM,TERM3 = KGG(4,1), in single precision complex expression,
CDTERM,TERM4 = KGG(4,1), in double precision complex expression
ATERM = 2nd word of the 2nd record of CASECC, integer, and
BTERM = 5th and 6th words of the 2nd record of CASECC, 2 BCD words.

APPENDIX G

USERS' MANUAL UPDATE PAGES FOR ADD MODULE

I. NAME: ADD (Matrix Add)

II. PURPOSE: To compute [X] = a[A] + b[B] where a and b are scale factors.

III. DMAP CALLING SEQUENCE:

```
ADD   A,B / X / C,Y, ALPHA=(1.0,2.0)      / C,Y, BETA=(3.0,4.0)
              / C,Y,DALPHA=(5.D+0,6.D-1) / C,Y,DBETA=(7.D+2,8.D-3)  $
```

IV. INPUT DATA BLOCKS:

A - Any GINO matrix

B - Any GINO matrix

V. OUTPUT DATA BLOCKS:

X - Matrix

VI. PARAMETERS:

ALPHA - Input-complex-single precision, default = (0.0, 0.0). This is a, the scalar multiplier for [A] if DALPHA and DBETA are zeros.
BETA - Input-complex-single precision, default = (0.0, 0.0). This is b, the scalar multiplier for [B] if DALPHA and DBETA are zeros.
DALPHA - Input-complex-double precision, default = (0.0D+0, 0.0D+0). This is a, the scalar multiplier for [A] if ALPHA and BETA are zeros.
DBETA - Input-complex-double precision, default = (0.0D+0, 0.0D+0). This is b, the scalar multiplier for [B] if ALPHA and BETA are zeros.

VII. SUBROUTINE: DADD

VIII. METHOD:

The parameters are checked. If [A] is not purged, the number of columns, rows, and form of [X] are set to those of [A]. Otherwise the [B] descriptors are used. The flags for the type of [X] (see Remark 2) and multiply-add operations are set before calling subroutine SADD, which performs the actual scalar multiplication and matrix addition.

VIII. REMARKS:

1. Matrix [A] and/or matrix [B] may be purged, in which case the corresponding term in the matrix sum will be assumed null. The input data blocks must be unique.

2. Matrix [X] cannot be purged. The type of [X] is maximum of the types of [A], [B], a, b. The size and shape of [X] are the size and shape of [A] if [A] is present. Otherwise they are those of [B].

3. The use of double precision parameters DALPHA and DBETA will force the matrix multiply-and-add operation to be performed in double precision unconditionally. The single precision ALPHA AND BETA may cause the multiply-and-add operation to be performed in single precision or in double precision depending on the matrix original precision types.

4. Either the DALPHA-DBETA pair or the ALPHA-BETA pair is used. They cannot be mixed; that is, DALPHA-BETA pair is illegal; so is DALPHA-ALPHA.

5. If Im(ALPHA or DALPHA) or Im(BETA or DBETA) is zero, the corresponding parameter will be considered real.

APPENDIX H

USERS' MANUAL UPDATE PAGES FOR THE GINOFILE MODULE

I. NAME: GINOFILE (Gino File Creation)

II. PURPOSE: To capture data from a scratch file of a preceding DMAP module and copy the data to a NASTRAN GINO file. Type of data can be table or matrix.

III. DMAP CALLING SEQUENCE:

```
GINOFILE  /FILE/C,N,P1/C,N,P2/C,N,P3   $
```

IV. INPUT DATA BLOCK: None.

V. OUTPUT DATA BLOCK:

FILE - Any GINO output file name

VI. PARAMETERS:

P1 - Any 300-series scratch File number (301,302,303,...), integer.

P2 - Additional records to be skipped on P1 file before data transfer from P1 to FILE, integer. GINOFILE will automatically skip over header record if a header record exists in P1, or it will not skip if it does not exist. (Default P2 = 0.) Data transfer starts from P2+1 record after header (or no header) record on scratch file.

P3 - Last record to be copied, or up to an EOF mark on P1 file. Total number of records copied is (P3 - P2), integer. (Default is to copy to EOF mark.)

VII. SUBROUTINE:

GINOFL - Subroutine in GINOFILE module

VIII. METHOD:

At the end of a NASTRAN executable module, all the input files, output files, and scratch files are closed. The input files are read only and they will remain untouched. The output files are saved, and their names are preserved. (The output file names are actually allocated before the beginning of the module execution). The scratch files are released without any mechanism of saving them. However, the data of the scratch files are still in the system disc space, and will remain there until they are over-written by another part (or another module) of the NASTRAN program. It is at this point that GINOFILE module grabs hold of a scratch file of the preceding module and copies the data to a GINO output file, without changing the scratch file data. Tables or matrices are copied the same way - as they exist in the original form on the scratch file.

A NASTRAN GINO file always has a header record and a 6 word trailer. However, the header record and the trailer are not required for a scratch file, and they may or may not exist. The GINOFILE module will first test the header record of the scratch file and skip over it, if it exists. A header record is always generated by GINOFILE for the new GINO file. The beginning record and the ending record where data are to be transferred are under user control. Finally, a trailer for the output file is generated and saved. An EOF record is written to the new GINO file at the completion of the module.

IX. DESIGN REQUIREMENT:

The GINOFILE module is mapped in all NASTRAN Links, except LINK1. The user can request this module through a regular NASTRAN DMAP Alter.

The user must request this module immediately following the DMAP module where the scratch file was used. It is the user's responsibility to see that the Executive Segment File Allocator, XSFA, does not come in between the preceding DMAP module and this GINOFILE module. If XSFA does intervene before GINOFILE execution, the FIAT/OSCAR table (see XSFA Module description in section 4.9) is rearranged, and the scratch files are no longer accessible.

If XSFA does intervene, the user can provoke the XSFA operation and FIAT/ OSCAR table rearrangement before the execution of preceding DMAP module so that XSFA will not come in between this preceding and GINOFILE modules. The technique here can involve a DMAP alter to PURGE some obsolete files, TABPT to print some files that have been generated some time ago, and currently are not on the FIAT/OSCAR table, or any other DMAP module that would disturb the NASTRAN filing system. The user could turn on DIAG 2 and observe the flow of the GINO files created or allocated by XSFA/FIAT/ OSCAR operation.

If the scratch file in the preceding DMAP module was used repeatedly such as being used in a loop, only the "last-time-used" set of data on the scratch file can be copied out by GINOFILE.

The user should turn on DIAG 8,15,-n (where n is the current LINK number) and see that the scratch file, FORTRAN unit number, and associated trailers are being processed correctly.

X. DIAGNOSTIC MESSAGES

Message numbers 3001, 3002, and 3008 may be issued by GINOFILE.

APPENDIX I

USERS' MANUAL UPDATE PAGES FOR THE NEW DATABASE MODULE

I. NAME: DATABASE (To Save Grids, Elements, Displacements, Velocities, Accelerations, Loads, Grid Point Forces, Eigenvectors, Element Stresses, and Element Forces on User Tape)

II. PURPOSE: To save following data on user tape, formatted, or unformatted for user external use:

(1) Grid points - external numbers, and their x,y,z coordinates in basic rectangular coordinate system;

(2) Connecting elements - element names, GPTABD element types, NASTRAN symbols, property IDs (or material IDs if elements have no property IDs), number of grid points, connecting grid (external) numbers; and

(3) Displacement vectors* - real or complex data in basic rectangular coordinate system, or in NASTRAN global coordinate system, in SORT1 or SORT2 data format, single-case or subcases, displacement or mode shape data.

(*including velocity, acceleration vectors, loads, grid point forces, eigenvectors, element stresses, and element forces)

III. DMAP CALLING SEQUENCE:

```
DATABASE   EQEXIN,BGPDT,GEOM2,CSTM,OUGV//C,N,OUTTP/C,N,FORMAT/C,N,BASIC  $
```

IV. INPUT DATA BLOCKS:

EQEXIN - External-internal grid tables. Must be present.

BGPDT - Basic Grid Point Definition Table.
If purge, no grid point data sent to OUTTP output tape.
If BGPDT is purged, and OUGV is present, displacement vector will not be converted to basic coordinates.

GEOM2 - Geometry 2 Data Block.
If purge, no element connectivity data sent to OUTTP.

CSTM - Coordinate System Transformation Matrix Data Block.
If purge, displacement vectors remain in global coordinate system.

OUGV - Any output displacement (velocity, acceleration, load, grid point force, eigenvector, element stress, and element force) data block written for OFP module. If present, the displacement vectors are processed and results sent out to user OUTTP tape. OUGV must be one of the following files characterized by an 1, 2, 3, 7, 10, 11, 15, or 16 on the 2nd word, last 2 digits, of the first header record, and an 8 or a 14 on the 10th word:

OUDV1, OUDVC1, OUGV1, OUHV1, OUHVC1, OUPV1, OUPVC1,
OUDV2, OUDVC2, OUGV2, OUHV2, OUHVC2, OUPV2, OUPVC2,
OUBGV1, OPHID, OPHIG, OPHIH, OCPHIP,
OPG1, OPP1, OPPC1, OQG1, OQP1, OQPC1, OQBG1,
OPG2, OPP2, OPPC2, OQG2, OQP2, OQPC2, OBQG1,
OEF1, OEFC1, OES1, OESC1, OEFB1, OBEF1, OEF2,
OEFC2, OES2, OESC2, OESB1, OBES1

If purge, no data are sent out to OUTTP.

V. <u>OUTPUT DATA BLOCK</u>: No GINO output data block.

VI. <u>PARAMETERS</u>:

OUTTP - User output tape. Must be one of the UT1, UT2, INPT, INP1, ..., INP9 files; tape or disc file. (Default INP1, FORTRAN Unit 15)

FORTRAN LOGICAL UNIT, OUTTP	USER FILE CODE
11	UT1 (CDC only)
12	UT2 (CDC only)
14	INPT (UNIVAC,VAX)
15	INP1 (All
16	INP2 machines
:	: except
23	INP9 CDC)
24	INPT (IBM only)

FORMAT = 0, unformatted output to OUTTP tape (Default);
= 1, formatted.

BASIC = 0, displacement vectors in NASTRAN's global coordinate system (Default);
= 1, displacement vectors in basic rectangular coordinate system.

VII. <u>EXAMPLE</u>:

```
DATABASE   EQEXIN,BGPDT,GEOM2,, /C,N,15/C,N,+1    $
DATABASE   EQEXIN,BGPDT,,CSTM,OUGV/C,N,16         $
```

First example writes the grid points and element connectivity data out to INP1 tape, formatted. The second example writes the grid points and displacement vectors in NASTRAN global coordinates out to INP2 tape, unformatted.

VIII. SUBROUTINE:

DBASE - Subroutine for DATABASE Module.

IX. METHOD:

There are three independent sets of data to be copied out to user tape OUTTP - grids data, connecting elements data, and displacement vectors (velocities, accelerations, eigenvectors, stresses, and forces). If BGPDT file is purged (that is, is not present), the grid point data set is not generated. Similarly, if GEOM2 file is purged, the element connectivity data is not generated; and the same with the OUGV file and the displacement vectors. The exact contents in the output tape OUTTP depend therefore on the input file assignment.

In all cases, EQEXIN file is opened and the grid point external number vs. the internal number table is read. If BGPDT file is present, the basic grid point data is read, and each internal grid point number is converted to its external ID number. The grid point's x, y, z coordinates from BGPDT are already in the basic rectangular coordinate system. The grid points data are then sorted by their external grid IDs before they are written out to OUTTP tape, under FORTRAN control. The following table gives the precise contents of each record in the OUTTP tape.

For UNFORMATTED tape - grid point data in one long record:

RECORD	WORD	CONTENT (UNFORMATTED)
1	1-2	'GRID PTS--------', a 16-letter identification. (BCD)
2	1	No. of words (this first word not included) in this record. (Integer)
	2	External grid ID. (Sorted, integer)
	3	0 (Not used; reserved for future use)
	4,5,6	x,y,z coordinates in basic rect. coord. system. (single precision real)
	:	Repeat words 2 thru 6 as many times as there are grids

(Total number of grid points = (WORD 1 of record 2)/5)

To read the second record into array XYZ, one can use

```
          READ (OUTTP) L,(XYZ(J),J=1,L)
```

For FORMATTED tape - grid point data in multiple short records:

RECORD	WORD	CONTENT	FORMAT
1	1,2	'GRID PTS--------' identification	4A4
2	1 .	Total number of grid points	I8
3	1	External grid ID (Sorted)	I8
	2	0 (Not used; Reserved for future use)	I8
	3,4,5	x,y,z coordinates in basic rect. coordinate system.	3E12.6
:	1-5	Repeat record 3 as many times as there are grids	

If GEOM2 file is present, the elements data will be generated next. An element identification record is written out first.

RECORD	WORD	CONTENT (FORMATTED or UNFORMATTED)	FORMAT
1	1-2	'ELEMENTS--------', identification. BCD	4A4

The element data in GEOM2 file will be written out to the OUTTP file almost in the same way, and same order as the original data. A header record is written out for each type of element, then followed by the element data. The element data will be written out in a long record if the OUTTP is unformatted, and in multiple short records, one for each element, if OUTTP is formatted. Notice that the element types are sorted according to the NASTRAN'S GPTABD data block order; and within each type, the elements are sorted by their element IDs.

ELEMENT HEADER RECORD for the UNFORMATTED output tape:

RECORD	WORD	CONTENT (UNFORMATTED)
2	1-2	Element name. (BCD)
	3	Element type number, according to GPTABD order. (Integer)
	4	Element symbol. (2 letters)
	5	Number of grid points per element. (Integer)
	6	Total no. of elements of this current element type. (Integer)
	7	No. of words in next record = WORD5 + 2 (Integer)
	8	No. of 132-column lines needed in next record if OUTTP is written with a format. (Integer)

ELEMENT RECORDS, Repeat as many times as there are elements not of the same type (that is a record for each element type):

RECORD	WORD	CONTENT (UNFORMATTED)
3	1	Element ID. (Integer)
	2	Property ID. (Positive Integer); or 0 (Element has no property ID nor material ID); or Material ID. (Element has no property ID, but it has a material ID. (Negative Integer)
	3	0 (Not used; Reserve for future use, integer)
	4,5,...	Element connecting (external) grid points. (Integers)
	:	Repeat words 1,2,3,4... as many times as there are elements of this same tape. (See WORD 6 in header record)

For FORMATTED tape -

ELEMENT HEADER RECORD, in 8-column format:

RECORD	COLUMNS	CONTENT	FORMAT
2	1- 8	'ELEMENT '	8 letters
	9-16	Element name	2A4
	17-24	' TYPE ='	8 letters
	25-28	Elem. type no. according to GPTABD	I4
	29,30	Blank	2X
	31-32	Element symbol	A2
	33-40	' GRIDS ='	8 letters
	41-48	No. of grids per element	I8
	49-56	' TOTAL ='	8 letters
	57-64	Total no. of elements of this elem. type	I8
	65-72	' WDS/EL='	8 letters
	73-80	No. of words per element in next records	I8
	81-88	' LINES ='	8 letters
	89-96	No. of lines (records) needed on next record for this element type	I8

A printout of this header record may look like this:
(the ---+++ line is for video aid; it is not part of the record)

```
 --------++++++++--------++++++++--------++++++++--------++++++++--
'ELEMENT CBAR      TYPE =  34  BR GRIDS =       2 TOTAL =      54 etc.'
```

ELEMENT RECORDS (FORMATTED) -
There should be (TOTAL X LINES) records in each element type:

RECORD	WORD	CONTENT	FORMAT
3	1	Element ID.	I8
	2	Property ID. (Positive integer); or 0 (Element has no property nor material ID); or Material ID. (Element has no property ID, but it has a material ID)	I8
	3	0 (Not used; reserve for future use)	I8
	4-16	First 13 external connecting grid points	13I8
4		(IF NEEDED, and LINES in header record = 2)	
	1-15	Next 15 Grid points	8X,15I8
5		(IF NEEDED, and LINES in header record = 3)	
	1-15	More grid points	8X,15I8
:	:	Repeat element record 3 (and possible 4 and 5) as many times as there are elements of the same type.	

Repeat the header record and the element records as many times as there are different types of elements.

The end of element data records is signaled by an element ENDING record of the following form, 8 words:

Words 1 and 2 form the word ' -END-',
Word 4 holds the symbol '--',
and all other words are zeros

The ENDING ELEMENT RECORD of the FORMATTED tape looks like this:

```
--------++++++++--------++++++++--------++++++++--------++++++++---
'ELEMENT -END-     TYPE =  0  -- GRIDS =       0 TOTAL =       0 etc.'
```

If the OUGV file is present, the displacement vectors will be processed and the final results sent out to the OUTTP tape. (In this and the next few paragraphs, the word "displacement" implies also velocity, acceleration, load, grid point force, eigenvector, element stresses, and element forces.) The input OUGV file must be one of the GINO files described in the INPUT DATA BLOCKS section, which gives the displacements in the g-set or p-set, or the other data types. The output data are sorted by their external grid ID numbers. The displacement records in OUTTP also begin with an identification record:

RECORD	WORD	CONTENT (FORMATTED or UNFORMATTED)	FORMAT
1	1-2	'DISPLCNT--------' identification*. BCD (* or 'VELOCITY--------', 'ACCELERN--------', 'LOADINGS--------', 'G FORCES--------', 'EIGENVCR--------', 'E STRESS--------', 'E FORCES--------'')	4A4

The original displacement data in NASTRAN are always in the global coordinate system. If the parameter BASIC is zero (default), the displacement vectors will be passed over to OUTTP without changes. However, if the parameter is set to +1, the displacement vectors will be converted to the basic rectangular coordinate system. In this latter case, the coordinate transformation matrices from CSTM will be brought into the computer, the grid point coordinate CID will be identified, and proper coordinate transformation will be applied to the displacements of each grid point. Again, the output OUTTP tape can be formatted or unformatted. In the unformatted tape, each grid point and its displacement values will form one logical record of 8 or 14 words (variable word length if element stresses or element forces). In the formatted tape, one logical record (8 words) is used if the displacement data is real, and an additional record (for data words 9 through 14) if the data is complex. In either case, a formatted record has 128-column of words. Similarly to the grid and element sets of data, a HEADER record is written out to OUTTP first before the grid point displacement vectors.

DISPLACEMENT HEADER RECORD for UNFORMATTED TAPE -

RECORD	WORD	CONTENT (UNFORMATTED)
2	1	Subcase or mode number. (Integer)
	2	Zero or frequency. (Real)
	3	Number of words per entry in next record.
	4-5	Original data file name, 2 BCD words
	6-7	' GLOBAL ' if BASIC=0, 2 BCD words ' BASIC ' if BASIC=1
	8-13	CODE (See note below; 6 integers)
	14-45	Title, 32 BCD words
	46-77	Subtitle, 32 BCD words
	78-109	Label, 32 BCD words

Note - Each code word holds 8 digits. Therefore there are 48 digits, from CODE(1) through CODE(6), and from left to right, they describe the data type of the next displacement record:

1 for integer
2 for real, and
3 for BCD

The first digit points to the first data word; 2nd, 3rd, 4th, etc. point to 2nd, 3rd, 4th data words, etc.

DISPLACEMENT RECORDS in UNFORMATTED tape - in one long record:

RECORD	WORD	CONTENT (UNFORMATTED)
3	1	No. of words (excluding this first word) in this record. (Integer)
	2	External grid point number. (Integer)
	3	Point type (1=grid pt. 2=scalar pt. 3=extra pt. 4=modal pt., integer)
	4-9	Displacements. (Real parts, t1,t2,t3,r1,r2,r3, single precision real)
	10-15	(COMPLEX data only) Displacements. (Imaginary parts, t1,t2,t3,r1,r2,r3, single precision real)
	:	Repeat words 2 thru 9 (or 15) as many times as there are grid points in OUGV file
:	:	Repeat record 3 as many times as there are subcases or frequencies

DISPLACEMENT HEADER RECORD for FORMATTED tape -

RECORD	WORD	CONTENT (FORMATTED)	FORMAT
2	1-2	' CASE = ' or ' MODE = '	8 letters
	3	Subcase number	I8
	4	Zero or frequency	1PE12.5
	5-6	' WORDS ='	8 letters
	7	NWDS, number of words per entry in next record (=8 for REAL data, or =14 COMPLEX, for all displacement records)	I8
	8-9	' INPUT ='	8 letters
	10-11	Original GINO file name	2A4
	12-13	' COORD ='	8 letters
	14-15	' BASIC ' or 'GLOBAL '	2A4
	16-17	' CODE ='	8 letters
	18-22	Format code 8 digits per word, 1 for INTEGER 2 for REAL Ex. 13222200 3 for BCD 0 not applicable	5I8
	23	NA4, number of words per entry in next record, in A4-word count	I8
3	1-32	Title, 32 BCD words	32A4
4	33-64	Subtitle, 32 BCD words	32A4
5	65-96	Label, 32 BCD words	32A4

DISPLACEMENT RECORDS in FORMATTED tape - in multiple short records:

RECORD	WORD	CONTENT	FORMAT
6	1	External grid point number. (Integer)	I8
	2	Point type (1=grid pt. 2=scalar pt. 3=extra pt. 4=modal pt., integer)	I8
	3-8	Displacements. (Real parts, t1,t2,t3,r1,r2,r3, single precision real)	6E12.6
7		(COMPLEX DATA only)	
	1-6	Displacements (Imaginary parts, t1,t2,t3,r1,r2,r3, single precision real)	16X,6E12.6
:	:	Repeat record 6 (records 6 and 7 if complex data) as many times as there are grid points displacement	

At the end of each subcase, if the output tape OUTTP is formatted, a ZERO record (two records if data is complex) is written out to OUTTP tape. This ZERO record has the same format as a DISPLACEMENT record, and consists of 8 or 14 zeros (first two are integers, minus zeros). This ZERO record is not needed in the unformatted OUTTP output tape.

Repeat the HEADER record, the DISPLACEMENT records, and the ZERO record (formatted OUTTP tape only) as many times as there are subcases. At the end of the last subcase, or end of the input file OUGV, an ENDING record is written out. It has the same form as the HEADER record:

DISPLACEMENT ENDING RECORD -

RECORD	WORD	CONTENT (UNFORMATTED)
LAST	1	Zero. (Integer)
	2	Zero. (Real)
	3	Zero. (Integer)
	4-5	' -END-'. (BCD)
	6-101	96 Blank words. (BCD)

RECORD	WORD	CONTENT (FORMATTED)	FORMAT
LAST	1-2	' CASE = ' or ' MODE = '	8-LETTERS
	3	Minus 0 (Integer)	I8
	4	Zero	1PE12.5
	5-6	' WORDS ='	8-LETTERS
	7	Minus 0 (Integer)	I8
	8-11	' INPUT = -END- '	16-LETTERS
	12-17	Blanks	4A4
LAST+1	1-32	Blanks	32A4
LAST+2	1-32	BLANKS	32A4
LAST+3	1-32	Blanks	32A4

If OUGV is an element stress or an element force file, the stress or force data have variable length depending on the type of element. The stress or force records written to the OUTTAP tape are therefore different from those of the displacement records.

THE ELEMENT STRESS or FORCE RECORD HAS the following forms:

RECORD	WORD	CONTENT (UNFORMATTED)
3	1	Number of words, excluding this first word, in this record. (Integer)
	2-NWDS	Element ID, stress or force data (Variable data types are described in 'CODE')
	:	Repeat (2-NWDS) words as many times as there are elements.
:	:	Repeat record 3 as many times as there are subcases.

where NWDS is the number of computer words per entry, and CODE is the 6-word format code, as described in header record.

or

RECORD	WORD	CONTENT (FORMATTED)	FORMAT
6	1-NA4	Element ID, stress or force data (The data types are described in 'CODE'; all integers in 2A4, real numbers in 3A4, and BCD in A4)	33A4
:	:	(Maximum record length is 132 columns (33A4); continuation into next record(s) if necessary)	
:	:	Repeat above record(s) as many times as there are elements	

where NA4 is the number of words per entry in A4-word count, and CODE is 5-word format code

Notice that the DATABASE module does not copy out the external-internal grid points table in EQEXIN file, nor the coordinate transformation matrices in CSTM. The coordinate systems originally associated with the external grid points are never mentioned in the OUTTP tape.

If the user must copy the EQEXIN and CSTM files (both are in table forms), the new OUTPUT5 can be used.

X. DESIGN REQUIREMENT:

The DATABASE module is mapped in NASTRAN Links 2, 4 and 14. This module is accessible only through a NASTRAN DMAP Alter.

Minimum open core requirement = 10 x (total number of grid points) words.

The formatted outputs are flagged only by the parameter FORMAT. The formatted output records are designed not to exceed 132 columns in length and include printer carriage control. In most cases, I8-formats are used for integers and E12.6 for real data (no double precision words used); and BCD words are in multiples of 2A4. The entire OUTTP file can be printed, or it can be edited by a system editor. The formatted OUTTP file, if written on magnetic tape by a computer, can be used in another computer of a different manufacturer.

The unformatted OUTTP file is more efficient; and the integer and real data are more accurate. The grid point data and data of each connecting element type are written out unformatted in long records; that requires large working space in the computer system. On the other hand, only short records are written to the formatted OUTTP file, and the working space requirement is less critical.

XI. REMARKS:

1. Conversion of element stresses or forces to the basic coordinates is not allowed.

XII. DIAGNOSTIC MESSAGES:

Message numbers 3001, 3002, and 3008 may be issued by DATABASE.

APPENDIX J

(A) A NASTRAN EXAMPLE USING DATABASE MODULE (OUTPUT LISTING SHORTENED)

```
NASTRAN   TITLEOPT=-1, FILES=INP1


                         ****
                       *      *
                      *        *
                      * N A S T R A N
                      *        *
                       *      *
                         ****


      DEC VAX COMPUTER SYSTEMS          SYSTEM RELEASE -  1989 ED.
      FTN VERSION -  50K

                             DISTRIBUTED BY

      COMPUTER SOFTWARE MANAGEMENT AND INFORMATION CENTER (COSMIC)
                           UNIVERSITY OF GEORGIA
                           ATHENS, GEORGIA 30602
                           PHONE  (404) 542-3265


    N A S T R A N   E X E C U T I V E   C O N T R O L   D E C K   E C H O

ID        TEST, VAX MACHINE
SOL       1,0
APP       DISP
ALTER     106
DATABASE  EQEXIN,BGPDT,GEOM2,CSTM,OUGV1 //C,N,15/C,N,+1/C,N,+1  $
ENDALTER
TIME 10
CEND


                                C A S E   C O N T R O L   D E C K   E C H O
          CARD
          COUNT
            1      TITLE    = TESTING DATABASE MODULE
            2      SUBTITLE = USING CYLINDRICAL COORDINATES
            3      LABEL    = GRIDS, ELEMENTS, AND DISPLACEMENTS OUTPUT TO INP1, FORMATTED
            4      SPC   = 10
            5      DISP  = ALL
            6      OLOAD = ALL
            7      ECHO  = BOTH
            8      SUBCASE 123
            9      LOAD  = 1000
           10      SUBCASE 456
```

```
 11         LOAD  = 2000
 12         BEGIN BULK
```

```
                         I N P U T   B U L K   D A T A   D E C K   E C H O

            ---1--- +++2+++ ---3--- +++4+++ ---5--- +++6+++ ---7--- +++8+++ ---9--- +++10+++
            -FF-    CORD1C, 3  101  333 999
            -FF-    GRID,101,, 10.  0.  0.,0, 123456
            -FF-    GRID,111,, 20.  0.  0.,0, 123456
            -FF-    GRID,222,, 30.  0.  0.,0, 123456
            -FF-    GRID,333,, 40.  0.  0.,0, 123456
            -FF-    GRID,555,, 20.  0. -9.+9,0, 123456
            -FF-    GRID,999,, 10. 10. 10.,0, 123456
            -FF-    GRDSET, 7)3
            -FF-    GRID,  1,  3,  5.     0.  0.
            -FF-    =(6),*(1), =,  =, *(15.), ==
            -FF-    GRID, 11,  3,  5.,    0., 10.
            -FF-    =(6),*(1), =,  =, *(15.), ==
            -FF-    GRID, 21,  3,  5.,    0., 20.
            -FF-    =(6),*(1), =,  =, *(15.), ==
            -FF-    GRID, 32,  3,  5.,   15., 30.
            -FF-    =(4),*(1), =,  =, *(15.), ==
            -FF-    GRID, 31,  0, 40.   3.53553  3.53553  0
            -FF-    GRID, 37,  0, 40.  -3.53553  3.53553  0
            $
            -FF-    CBAR,  1,2, 1    2, 101
            -FF-    =(5),*(1),=, *(1),/,=
            -FF-    CBAR, 11,2, 11  12, 111
            -FF-    =(5),*(1),=, *(1),/,=
            -FF-    CBAR, 21,2, 21  22, 222
            -FF-    =(5),*(1),=, *(1),/,=
            -FF-    CBAR, 31,2, 31  32, 333
            -FF-    =(5),*(1),=, *(1),/,=
            -FF-    CBAR, 41,2,  1  11, 555
            -FF-    =(2),*(1),=, *(10),/,=
            -FF-    CBAR, 51,2,  7  17, 555
            -FF-    =(2),*(1),=, *(10),/,=
            $
            -FF-    CQUAD2,71, 7,  1  11  12   2
            -FF-    =(5),*(1), =,*(1),///
            -FF-    CQUAD2,81, 7, 11  21  22  12
            -FF-    =(5),*(1), =,*(1),///
            -FF-    CQUAD2,91, 7, 21  31  32  22
            -FF-    =(5),*(1), =,*(1),///
            $
            -FF-    PBAR,  2,100, .4  .5  .3  .3
            -FF-    PQUAD2,7,100, .05
            -FF-    MAT1,100  3.0E+7,, .3 1.0
            -FF-    SPC1,10,123456, 1 THRU 7
            -FF-    FORCE,1000,31,0,100.0,  0.0, 0., -1.0
            -FF-    FORCE,1000,37,0,100.0,  0.0, 0., -1.0
            -FF-    FORCE,2000,34,3,200.0, -1.0, 0.   0.0
            ENDDATA
      TOTAL COUNT=     46
```

```
                              S O R T E D   B U L K   D A T A   E C H O
CARD
COUNT      ---1--- +++2+++ ---3--- +++4+++ ---5--- +++6+++ ---7--- +++8+++ ---9--- +++10+++
    1-     CBAR    1       2       1       2       101
    2-     CBAR    2       2       2       3       101
    :
    5-     CBAR    5       2       5       6       101
    6-     CBAR    6       2       6       7       101
    7-     CBAR    11      2       11      12      111
    8-     CBAR    12      2       12      13      111
    9-     CBAR    13      2       13      14      111
    :
   17-     CBAR    25      2       25      26      222
   18-     CBAR    26      2       26      27      222
   19-     CBAR    31      2       31      32      333
   20-     CBAR  . 32      2       32      33      333
    :
   24-     CBAR    36      2       36      37      333
   25-     CBAR    41      2       1       11      555
   26-     CBAR    42      2       11      21      555
   27-     CBAR    43      2       21      31      555
   28-     CBAR    51      2       7       17      555
   29-     CBAR    52      2       17      27      555
   30-     CBAR    53      2       27      37      555
   31-     CORD1C  3       101     333     999
   32-     CQUAD2  71      7       1       11      12      2
   33-     CQUAD2  72      7       2       12      13      3
    :
   37-     CQUAD2  76      7       6       16      17      7
   38-     CQUAD2  81      7       11      21      22      12
   39-     CQUAD2  82      7       12      22      23      13
    :
   43-     CQUAD2  86      7       16      26      27      17
   44-     CQUAD2  91      7       21      31      32      22
   45-     CQUAD2  92      7       22      32      33      23
    :
   48-     CQUAD2  95      7       25      35      36      26
   49-     CQUAD2  96      7       26      36      37      27
   50-     FORCE   1000    31      0       100.0   0.0     0.      -1.0
   51-     FORCE   1000    37      0       100.0   0.0     0.      -1.0
   52-     FORCE   2000    34      3       200.0   -1.0    0.      0.0
   53-     GRDSET                                          3
   54-     GRID    1       3       5.      0.      0.
   55-     GRID    2       3       5.      15.     0.
   56-     GRID    3       3       5.      30.     0.
   57-     GRID    4       3       5.      45.     0.
   58-     GRID    5       3       5.      60.     0.
   59-     GRID    6       3       5.      75.     0.
   60-     GRID    7       3       5.      90.     0.
   61-     GRID    11      3       5.      0.      10.
   62-     GRID    12      3       5.      15.     10.
   63-     GRID    13      3       5.      30.     10.
    :
   73-     GRID    26      3       5.      75.     20.
   74-     GRID    27      3       5.      90.     20.
```

```
75-      GRID    31      0       40.     3.53553 3.53553 0
76-      GRID    32      3       5.      15.     30.
77-      GRID    33      3       5.      30.     30.
78-      GRID    34      3       5.      45.     30.
79-      GRID    35      3       5.      60.     30.
80-      GRID    36      3       5.      75.     30.
8*       GRID    37      0       40.     -3.535533.53553 0
82-      GRID    101             10.     0.      0.      0       123456
83-      GRID    111             20.     0.      0.      0       123456
84-      GRID    222             30.     0.      0.      0       123456
 :
87-      GRID    999             10.     10.     10.     0       123456
88-      MAT1    100     3.0E+7          .3      1.0
89-      PBAR    2       100     .4      .5      .3      .3
90-      PQUAD2  7       100     .05
91-      SPC1    10      123456  1       THRU    7
         ENDDATA
```

```
*** USER INFORMATION MESSAGES FROM RESEQUENCING PROCESSOR - BANDIT    (CRI= 1,  MTH= 3,  MPC= 0,  DEP=-1,  PCH=-1)

                      BEFORE RESEQUENCING - - -
                            BANDWIDTH            9
                                :

                      AFTER RESEQUENCING BY GIBBS-POOLE-STOCKMEYER (GPS) ALGORITHM - - -
                            BANDWIDTH            6
                                :
                            RMS WAVEFRONT    5.335


                                          *** BANDIT SUMMARY ***

                                                              BEFORE     AFTER
                            BANDWIDTH (B)                          9         6
                            PROFILE (P)                          199       145
                            MAXIMUM WAVEFRONT (C-MAX)              9         6
                            AVERAGE WAVEFRONT (C-AVG)          7.107     5.179
                            RMS WAVEFRONT (C-RMS)              7.500     5.335
                            NUMBER OF GRID POINTS (N)                       34
                            NUMBER OF ELEMENTS (NON-RIGID)                  48
                            NUMBER OF RIGID ELEMENTS PROCESSED*              0
                                  :
                            CRITERION*                           RMS WAVEFRONT
                            METHOD USED*                                   GPS
                            NO. OF NON-ACTIVE GRID POINTS                    6
                            NO. OF SEQGP CARDS GENERATED                     9


                      S Y S T E M   G E N E R A T E D   S E Q G P   C A R D S

                 SEQGP         1       1       2       5       3       9       4      13
                 SEQGP         5      17       6      21       7      25      11       2
```

```
                            :
                        SEQGP         101       29      111       30      222       31      333       32
                        SEQGP         555       33      999       34
         **NO ERRORS FOUND - EXECUTE NASTRAN PROGRAM**


*** USER INFORMATION MESSAGE 3035
FOR SUBCASE NUMBER     1, EPSILON SUB E =  8.3046791E-13
*** USER INFORMATION MESSAGE 3035
FOR SUBCASE NUMBER     2, EPSILON SUB E = -2.6844558E-13


*** USER INFORMATION MESSAGE -
    DATABASE MODULE TRANSFERRED THE FOLLOWING  3 SETS OF DATA TO OUTPUT FILE  INP1   (FORTRAN UNIT 15),   FORMATTED

    1. GRID POINT DATA - EXTERNAL NUMBERS AND BASIC RECTANGULAR COORDINATES

    2. ELEMENT CONNECTIVITY DATA - ALL GRIDS POINTS ARE EXTERNAL NUMBERS

    3. DISPLCNT DATA FROM INPUT FILE OUGV1   , DATA CONVERTED TO BASIC RECT. COORDINATES,    2 SUBCASES


     TESTING DATABASE MODULE                                                      JANUARY  19, 1989    RELEASE  1989  VAX
     USING CYLINDRICAL COORDINATES
     GRIDS, ELEMENTS, AND DISPLACEMENTS OUTPUT TO INP1, FORMATTED                                              SUBCASE 123


                                             D I S P L A C E M E N T   V E C T O R

     POINT ID.   TYPE          T1              T2              T3              R1              R2              R3
            1      G     0.0             0.0             0.0             0.0             0.0             0.0
            2      G     0.0             0.0             0.0             0.0             0.0             0.0
            :
            7      G     0.0             0.0             0.0             0.0             0.0             0.0
           11      G    -3.318057E-03   -3.292276E-03   -2.601234E-04    5.895234E-04   -6.161431E-04    4.702619E-06
           12      G    -4.063191E-03   -2.321371E-03    3.838165E-04    3.955249E-04   -7.332436E-04    4.287300E-06
           13      G    -4.531379E-03   -1.199775E-03    7.735121E-04    1.998269E-04   -7.991001E-04    2.540319E-06
            :
           17      G    -3.318057E-03    3.292276E-03   -2.601235E-04   -5.895234E-04   -6.161430E-04   -4.702721E-06
           21      G    -1.135364E-02   -1.133029E-02   -4.189800E-04    9.427494E-04   -9.638459E-04   -2.057090E-06
           22      G    -1.389689E-02   -8.006683E-03    6.217674E-04    6.450522E-04   -1.161740E-03   -1.690856E-06
            :
           36      G    -2.659377E-02    1.533314E-02    7.031241E-04   -7.324436E-04   -1.305977E-03    1.251280E-05
           37      G    -4.714102E-04    2.952591E-05   -3.071109E-02    1.414761E-05    1.510598E-03    1.724372E-05
          101      G     0.0             0.0             0.0             0.0             0.0             0.0
          111      G     0.0             0.0             0.0             0.0             0.0             0.0
            :
          999      G     0.0             0.0             0.0             0.0             0.0             0.0
```

```
TESTING DATABASE MODULE                                        JANUARY  19, 1989   RELEASE  1989  VAX
USING CYLINDRICAL COORDINATES
GRIDS, ELEMENTS, AND DISPLACEMENTS OUTPUT TO INP1, FORMATTED                                SUBCASE 456

                                 D I S P L A C E M E N T   V E C T O R

POINT ID.   TYPE          T1             T2             T3             R1             R2             R3
        1     G      0.0            0.0            0.0            0.0            0.0            0.0
        2     G      0.0            0.0            0.0            0.0            0.0            0.0
        :
        7     G      0.0            0.0            0.0            0.0            0.0            0.0
       11     G     -3.318148E-03  -3.292200E-03  -2.601124E-04   5.894569E-04  -6.161953E-04   4.774461E-06
       12     G     -4.063299E-03  -2.321291E-03   3.838149E-04   3.955561E-04  -7.332786E-04   4.285285E-06
        :
       16     G     -4.063299E-03   2.321291E-03   3.838153E-04  -3.955560E-04  -7.332786E-04  -4.285384E-06
       17     G     -3.318147E-03   3.292200E-03  -2.601125E-04  -5.894569E-04  -6.161953E-04  -4.774563E-06
       21     G     -1.134697E-02  -1.132912E-02  -4.194220E-04   9.431769E-04  -9.614227E-04   1.676048E-06
       22     G     -1.389428E-02  -8.006698E-03   6.221989E-04   6.458150E-04  -1.160364E-03   9.538337E-07
        :
       27     G     -1.134696E-02   1.132912E-02  -4.194221E-04  -9.431769E-04  -9.614226E-04  -1.676340E-06
       31     G     -4.712943E-04   1.271940E-05  -3.068326E-02   2.592916E-05   1.508337E-03  -1.354597E-05
        :
       37     G     -4.712941E-04  -1.271908E-05  -3.068326E-02  -2.592964E-05   1.508337E-03   1.354600E-05
      101     G      0.0            0.0            0.0            0.0            0.0            0.0
        :
      555     G      0.0            0.0            0.0            0.0            0.0            0.0
      999     G      0.0            0.0            0.0            0.0            0.0            0.0


TESTING DATABASE MODULE                                        JANUARY  19, 1989   RELEASE  1989  VAX
USING CYLINDRICAL COORDINATES
GRIDS, ELEMENTS, AND DISPLACEMENTS OUTPUT TO INP1, FORMATTED                                SUBCASE 123

                                         L O A D   V E C T O R

POINT ID.   TYPE          T1             T2             T3             R1             R2             R3
       31     G      0.0            0.0           -1.000000E+02   0.0            0.0            0.0
       37     G      0.0            0.0           -1.000000E+02   0.0            0.0            0.0

TESTING DATABASE MODULE                                        JANUARY  19, 1989   RELEASE  1989  VAX
USING CYLINDRICAL COORDINATES
GRIDS, ELEMENTS, AND DISPLACEMENTS OUTPUT TO INP1, FORMATTED                                SUBCASE 456

                                         L O A D   V E C T O R

POINT ID.   TYPE          T1             T2             T3             R1             R2             R3
       34     G     -2.000000E+02   0.0            0.0            0.0            0.0            0.0

                                  * * * END OF JOB * * *
```

(B) FORMATTED INP1 FILE AS GENERATED FROM ABOVE NASTRAN RUN (LISTING SHORTENED)

```
GRID PTS--------
     34= TOTAL NUMBER OF GRID POINTS
      1       0 1.00000E+01 3.53553E+00 3.53553E+00
      2       0 1.00000E+01 2.50000E+00 4.33013E+00
      3       0 1.00000E+01 1.29410E+00 4.82963E+00
      4       0 1.00000E+01 0.00000E+00 5.00000E+00
      5       0 1.00000E+01-1.29410E+00 4.82963E+00
      6       0 1.00000E+01-2.50000E+00 4.33013E+00
      7       0 1.00000E+01-3.53553E+00 3.53553E+00
     11       0 2.00000E+01 3.53553E+00 3.53553E+00
     12       0 2.00000E+01 2.50000E+00 4.33013E+00
      :
     26       0 3.00000E+01-2.50000E+00 4.33013E+00
     27       0 3.00000E+01-3.53553E+00 3.53553E+00
     31       0 4.00000E+01 3.53553E+00 3.53553E+00
     32       0 4.00000E+01 2.50000E+00 4.33013E+00
      :
     37       0 4.00000E+01-3.53553E+00 3.53553E+00
    101       0 1.00000E+01 0.00000E+00 0.00000E+00
    111       0 2.00000E+01 0.00000E+00 0.00000E+00
      :
    555       0 2.00000E+01 0.00000E+00-9.00000E+09
    999       0 1.00000E+01 1.00000E+01 1.00000E+01
ELEMENTS--------
ELEMENT BAR        TYPE =  34  BR GRIDS =       2 TOTAL =       30 WDS/EL=      5 LINES =       1
      1      2       0       1       2
      2      2       0       2       3
      3      2       0       3       4
      4      2       0       4       5
      5      2       0       5       6
      6      2       0       6       7
     11      2       0      11      12
     12      2       0      12      13
      :
     25      2       0      25      26
     26      2       0      26      27
     31      2       0      31      32
     32      2       0      32      33
      :
     36      2       0      36      37
     41      2       0       1      11
     42      2       0      11      21
      :
     53      2       0      27      37
ELEMENT QUAD2      TYPE =  18  Q2 GRIDS =       4 TOTAL =       18 WDS/EL=      7 LINES =       1
     71      7       0       1      11      12       2
     72      7       0       2      12      13       3
     73      7       0       3      13      14       4
     74      7       0       4      14      15       5
     75      7       0       5      15      16       6
     76      7       0       6      16      17       7
     81      7       0      11      21      22      12
```

```
      82        7        0       12       22       23       13
       :
      95        7        0       25       35       36       26
      96        7        0       26       36       37       27
ELEMENT  -END-    TYPE =   0  -- GRIDS =        0 TOTAL =       0 WDS/EL=       0 LINES =        0
DISPLCNT--------
 CASE =       123 0.00000E+00 WORDS =        8 INPUT =OUGV1    COORD =  BASIC   CODE = 11222222
 TESTING DATABASE MODULE
 USING CYLINDRICAL COORDINATES
 GRIDS, ELEMENTS, AND DISPLACEMENTS OUTPUT TO INP1, FORMATTED                               SUBCASE 123
       1        1 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
       2        1 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
       :
       7        1 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
      11        1-2.60123E-04-1.82304E-05-4.67421E-03 4.70262E-06 8.52535E-04-1.88229E-05
      12        1 3.83816E-04-2.12288E-05-4.67951E-03 4.28730E-06 8.32770E-04-2.40871E-05
      13        1 7.73512E-04-1.39135E-05-4.68750E-03 2.54032E-06 8.23590E-04-1.38043E-05
       :
      17        1-2.60124E-04 1.82304E-05-4.67421E-03-4.70272E-06 8.52535E-04 1.88230E-05
      21        1-4.18980E-04-1.65105E-05-1.60400E-02-2.05709E-06 1.34817E-03-1.49174E-05
      22        1 6.21767E-04-1.44541E-05-1.60384E-02-1.69086E-06 1.32862E-03-2.22383E-05
       :
      36        1 7.03124E-04 1.79987E-05-3.06975E-02 1.25128E-05 1.49723E-03 1.86739E-05
      37        1-4.71410E-04 2.95259E-05-3.07111E-02 1.41476E-05 1.51060E-03 1.72437E-05
     101        1 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
     111        1 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
       :
     999        1 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
      -0       -0 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
 CASE =       456 0.00000E+00 WORDS =        8 INPUT =OUGV1    COORD =  BASIC   CODE = 11222222
 TESTING DATABASE MODULE
 USING CYLINDRICAL COORDINATES
 GRIDS, ELEMENTS, AND DISPLACEMENTS OUTPUT TO INP1, FORMATTED                               SUBCASE 456
       1        1 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
       2        1 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
       :
       7        1 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
      11        1-2.60112E-04-1.83478E-05-4.67422E-03 4.77446E-06 8.52525E-04-1.89070E-05
      12        1 3.83815E-04-2.13520E-05-4.67957E-03 4.28529E-06 8.32816E-04-2.40777E-05
       :
      16        1 3.83815E-04 2.13524E-05-4.67957E-03-4.28538E-06 8.32816E-04 2.40777E-05
      17        1-2.60113E-04 1.83480E-05-4.67422E-03-4.77456E-06 8.52525E-04 1.89070E-05
      21        1-4.19422E-04-1.26166E-05-1.60344E-02 1.67605E-06 1.34676E-03-1.29017E-05
      22        1 6.22199E-04-1.31377E-05-1.60362E-02 9.53834E-07 1.32781E-03-2.08896E-05
       :
      27        1-4.19422E-04 1.26157E-05-1.60344E-02-1.67634E-06 1.34676E-03 1.29018E-05
      31        1-4.71294E-04 1.27194E-05-3.06833E-02 2.59292E-05 1.50834E-03-1.35460E-05
       :
      37        1-4.71294E-04-1.27191E-05-3.06833E-02-2.59296E-05 1.50834E-03 1.35460E-05
     101        1 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
       :
     555        1 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
     999        1 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
      -0       -0 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
 CASE =         0 0.00000E+00 WORDS =        0 INPUT = -END-   COORD =           CODE =        0
```

(C) RDBASE - A FORTRAN PROGRAM TO READ UNFORMATTED OUTPUT FILE GENERATED BY DATABASE MODULE

```
      PROGRAM RDBASE
C
C     THIS FORTRAN PROGRAM READS THE UNFORMATTED OUTPUT FILE INP1
C     (FORTRAN UNIT 15) GENERATED BY DATABASE MODULE
C
C     (1) GRID POINTS DATA ARE READ AND SAVED IN GRID-ARRAY
C     (2) ELEMENTS DATA ARE READ AND SAVED IN ELM-ARRAY,
C         WITH ELEMENT NAMES AND POINTERS IN SAVE-ARRAY
C     (3) DISPLACEMENTS (VELOCITIES, ACCELERATIONS, LOADS, GRID-POINT
C         FORCE, OR EIGENVECTORS) DATA ARE READ AND SAVED IN DIS-ARRAY,
C         WITH SUBASES AND POINTERS IN SAVD-ARRAY
C
C     ANY OF ABOVE 3 SETS OF DATA NEED NOT EXIST IN ORIGINAL INP1 FILE
C
C     TO READ ELEMENT FORCES OR ELEMENT STRESSES, (3) ABOVE NEEDS SOME
C     CHANGES. PARTICULARLY WE NEED THE INFORMATION IN CODE TO GIVE US
C     THE TYPE OF EACH DATA WORD IN THE DATA LINE.
C     ASSUME CODE(1) = 11222222
C            CODE(2) = 31222000
C     THIS MEANS
C            THE 1ST, 2ND, AND 10TH DATA WORDS ARE INTEGERS;
C            9TH DATA WORD IS BCD; AND
C            3RD THRU 8TH, 11TH, 12TH AND 13TH WORDS ARE REAL NUMBERS
C
C
C     WRITTEN BY G.CHAN/UNISYS, JAN. 1989
C
      IMPLICIT INTEGER (A-Z)
      INTEGER          GRID(5,500),ELM(35,300),DIS(11200),SAVE(4,10),
     1                 SAVD(3,20),NAME(2),TITLE(32),SUBTTL(32),
     2                 LABL(32),CODE(6)
      REAL             GRIR(5,1),RIS(1),FREQ
      DOUBLE PRECISION GED,GD,EL,DS,ENDO,COORD
      EQUIVALENCE      (GRID(1),GRIR(1)),(DIS(1),RIS(1))
      DATA             INTAP, NOUT, MAXGRD, MAXELM, MAXDIS, MAXWDS    /
     1                 15,    6,     500,    300,    11200,  35       /
      DATA             GD,           EL,           DS,           END1 /
     1                 8HGRID PTS, 8HELEMENTS, 8HDISPLCNT,  4H -EN    /
C
      REWIND INTAP
C
C     READ DATA IDENTICATION RECORD
C
 70   READ (INTAP,END=400) GED
      IF (NOUT .EQ. 6) WRITE (NOUT,80) GED
 80   FORMAT (1X,A8,'---------')
      IF (GED .EQ. GD) GO TO 100
      IF (GED .EQ. EL) GO TO 200
      IF (GED .EQ. DS) GO TO 300
      STOP 'DATA TYPE UNKNOWN'
C
```

```
C     PROCESS GRID DATA
C     =================
C
C     READ GRID POINT DATA, ONE LONG RECORD OF MIXED INTEGERS AND REALS
C
 100  READ (INTAP,END=400) L,(GRID(J,1),J=1,L)
      IF (NOUT .NE. 6) GO TO 70
      NGRID = L/5
      IF (NGRID .GT. MAXGRD) STOP 'GRID DIMENSION TOO SMALL'
      WRITE (NOUT,120) NGRID
 120  FORMAT (1X,I8,'=TOTAL NO. OF GRID POINTS')
      DO 140 I=1,NGRID
      WRITE (NOUT,130) GRID(1,I),GRID(2,I),GRIR(3,I),GRIR(4,I),GRIR(5,I)
 130  FORMAT (1X,2I8,3(1PE12.5))
 140  CONTINUE
      GO TO 70
C
C     PROCESS ELEMENT DATA
C     ====================
C
 200  JS = 0
      JE = 0
C
C     READ ELEMENT HEADER RECORD, 8 WORDS
C
 210  READ (INTAP,END=400) NAME,TYPE,SYMBOL,GRIDS,TOTAL,WDS,LINE
      IF (NAME(1).EQ.END1 .AND. TYPE.EQ.0) GO TO 250
      IF (WDS .GT. MAXWDS) STOP 'ELM ROW DIMENSION TOO SMALL'
      IF (JE  .GT. MAXELM) STOP 'ELM COL DIMENSION TOO SMALL'
      JB = JE+1
      JE = JE+TOTAL
C
C     READ ELEMENT DATA, ONE LONG RECORD PER ELEMENT TYPE (ALL INTEGERS)
C
      READ (INTAP) ((ELM(I,J),I=1,WDS),J=JB,JE)
      JS = JS+1
      IF (JS .GE. 10) STOP 'SAVE DIMENSION TOO SMALL'
C
C     SAVE ELEMENT NAMES AND BEGINNING POINTERS IN SAVE-ARRAY
C     FOR EASY IDENTIFICATION
C
      SAVE(1,JS) = NAME(1)
      SAVE(2,JS) = NAME(2)
      SAVE(3,JS) = JB
      SAVE(4,JS) = WDS
      IF (NOUT .NE. 6) GO TO 210
      WRITE (NOUT,220) NAME,TYPE,SYMBOL,GRIDS,TOTAL,WDS,LINE
 220  FORMAT (1X,'ELEMNT =',2A4,'  TYPE =',I4,2X,A2,' GRIDS =',I8,
     1           ' TOTAL =',I8,' WDS/EL=',I8,       ' LINE  =',I8)
      DO 240 J=JB,JE
      WRITE (NOUT,230) (ELM(I,J),I=1,WDS)
 230  FORMAT (1X,3I8,13I8, /,(1X,8X,15I8))
 240  CONTINUE
      GO TO 210
C
```

BEST COPY
AVAILABLE

```
C     WRAP UP SAVE-ARRAY
C
 250  JS = JS+1
      SAVE(1,JS) = END1
      SAVE(2,JS) = NAME(2)
      SAVE(3,JS) = JE+1
      SAVE(4,JS) = 0
      IF (NOUT .NE. 6) GO TO 70
      WRITE (NOUT,260)
      WRITE (NOUT,270) ((SAVE(I,J),I=1,4),J=1,JS)
 260  FORMAT (/30X,'THIS REFERENCE TABLE IS NOT PART OF INPUT FILE')
 270  FORMAT (40X,2A4,3H @ ,I4,',  WORDS=',I3)
      GO TO 70
C
C     PROCESS DISPLACEMENT DATA
C     ************************
C
C
 290  STOP 'ERROR IN READING DISPLACEMENT DATA'
C
 300  KB = 1
      KS = 0
C
C     READ DISPLACEMENT HEADER RECORD
C
 310  KS = KS+1
      IF (KS .GT. 20) STOP 'SAVD DEMINSION TOO SMALL'
      READ (INTAP,END=380) CASE,FREQ,NWDS,NAME,COORD,CODE,TITLE,SUBTTL,
     1                     LABEL
      IF (CASE+NWDS .EQ. 0) GO TO 380
      IF (NOUT .NE. 6) GO TO 330
      WRITE (NOUT,320) CASE,FREQ,NWDS,NAME,COORD,CODE(1),CODE(2),TITLE,
     1                 SUBTTL,LABEL
 320  FORMAT ('  CASES =',I8,1PE12.5,' WORDS =',I8,' INPUT =',2A4,
     1       '  COORD =',A8,'  CODE = ',2I8, /,(1X,32A4))
C
C     DISPLACEMENT RECORS HAVE EITHER 8 OR 14 WORDS EACH DATA POINT
C     WITH CODE(1)=11222222, CODE(2) THRU (5) ARE ZEROS.
C
C     (IF THIS WERE TO READ ELEMENT STRESS RECORDS, THERE WOULD BE
C     NWDS DATA WORDS PER ELEMENT, AND THERE WOULD BE NWDS DIGITS IN
C     CODE INDICATING THE TYPE OF EACH DATA WORD. FIRST DIGIT (FROM
C     LEFT TO RIGHT) POINTS TO THE DATA TYPE OF FIRST DATA WORD,
C     2ND DIGIT TO 2ND DATA WORD, AND SO ON. SEE EXAMPLE AT THE
C     BEGINNING OF THIS SUBROUTINE)
C
 330  IF (NWDS.NE.8 .AND. NWDS.NE.14) STOP 'WORD COUNT ERROR'
      IF (CODE(1) .NE. 11222222) STOP 'FORMAT CODE ERROR'
C
C     SAVE SUBCASE NUMBER AND BEGINNING POINTERS IN SAVD-ARRAY
C     FOR EASY IDENTIFICATION
C
      KBM1 = KB-1
      SAVD(1,KS) = CASE
      SAVD(2,KS) = KB
```

BEST COPY
AVAILABLE

```
      SAVD(3,KS) = NWDS
C
C     READ DISPLACEMENT RECORD, ONE LONG RECORD PER SUBCASE (OR FREQ.)
C     EACH GRID POINT DISPLACEMENT DATA IN EVERY 8 OR 14 WORDS,
C     2 INTEGERS + 6 (OR 12) REALS
C
 340  READ (INTAP,ERR=290) L,(DIS(I+KBM1),I=1,L)
      KE = L+KBM1
      DO 370 K=KB,KE,NWDS
      WRITE (NOUT,350)  DIS(K),DIS(K+1), (RIS(K+I),I=2, 7)
      IF (NWDS .EQ. 14) WRITE (NOUT,360) (RIS(K+I),I=8,13)
 350  FORMAT (1X,2I8,6(1PE12.5))
 360  FORMAT (1X,16X,6(1PE12.5))
 370  CONTINUE
      KB = KE+1
      GO TO 310
C
C     WRAP UP SAVD-ARRAY
C
 380  SAVD(1,KS) = 0
      SAVD(2,KS) = KE+1
      SAVD(3,KS) = 0
      IF (NOUT .NE. 6) GO TO 70
      WRITE (NOUT,260)
      WRITE (NOUT,390) (SAVD(1,K),SAVD(2,K),SAVD(3,K),K=1,KS)
 390  FORMAT (40X,'CASE',I8,3H @ ,I4,',  WORDS=',I4)
      GO TO 70
C
 400  REWIND INTAP
      END
```

BEST COPY
AVAILABLE

APPENDIX K

USERS' MANUAL UPDATE PAGES FOR THE NEW RIGID ELEMENTS

Input Data Card CRROD Rigid Pin-Ended Rod

Description: Defines a pin-ended rod that is rigid in extension-compression.

Format and Example:

1	2	3	4	5	6	7	8	9	10
CRROD	EID	G1	G2	C1	C2				
CRROD	14	1	2	2					

Field	Contents
EID	Element identification number (Integer > 0)
Gi	Identification numbers of connection grid points (Integers > 0)
Ci	Component number of one and only one dependent translational degree of freedom in the global coordinate system assigned to either G1 or G2. (Integer equals to 1, 2, or 3.) Either C1 or C2 must contain an integer and the other must be blank. See Remarks 2 and 3.

Remarks:

1. Element identification number must be unique with respect to all other element identification numbers.

2. The grid point that associates with a blank Ci field, is designated as the reference independent grid point.

3. The dependent (that is constrained) degrees of freedom in a CRROD element may not appear on OMIT, OMIT1, SPC, or SUPORT cards, nor may they be redundantly implied on ASET or ASET1 cards. They may not appear as dependent degrees of freedom in other rigid elements or on MPC cards. Degrees of freedom declared to be independent by a rigid element can be made dependent by another rigid element or by an MPC card.

4. Rigid elements, unlike MPC's, are not selected through the Case Control Deck.

5. Forces of constraint are not recovered.

6. Rigid elements are ignored in heat transfer problems.

7. The degree of freedom selected to be dependent must have a nonzero component along the axis of the rod.

8. Nastran actually converts the CRROD input card into the CRIGDR card format, and thus processes a CRROD card as if it were a CRIGDR card. The following table shows the conversion, in free-field format, of two possible cases:

Case	CRROD Card	Equivalent CRIGDR Card
1	CRROD, EID, G1, G2, C1,	CRIGDR, EID, G2, G1, C1
2	CRROD, EID, G1, G2, , C2	CRIGDR, EID, G1, G2, C2

9. See section 1.4.2.2 for a discussion of rigid elements.

Input Data Card CRBAR Rigid Bar

Description: Defines a rigid bar with six degrees of freedom at each end.

Format and Example:

1	2	3	4	5	6	7	8	9	10
CRBAR	EID	G1	G2	IC1	IC2	DC1	DC2		
CRBAR	5	1	2	234	123				

Field	Contents
EID	Element identification number (Integer > 0)
Gi	Identification numbers of connection grid points (Integers > 0)
ICi	Independent degrees of freedom in the global coordinate system for the element at grid points Gi (any of the digits 1-6 with no imbedded blanks. Integers ⩾ 0 or blank.) See Remark 2.
DCi	Dependent degrees of freedom in the global coordinate system assigned by the element at grid points Gi (any of the digits 1-6 with no imbedded blanks. Integers ⩾ 0 or blank.) See Remarks 3 and 4.

Remarks:

1. Element identification number must be unique with respect to all other element identification numbers.

2. The total number of degrees of freedom specified (IC1 and IC2) must equal six; for example, IC1 = 1236, IC2 = 34. Further, they should together be capable of representing any general rigid body motion of the element.

3. If both DC1 and DC2 are zero or blank, all of the degrees of freedom not in IC1 and IC2 will be made dependent.

4. The dependent (that is, constrained) degrees of freedom in a CRBAR element may not appear on OMIT, OMIT1, SPC, or SUPORT cards, nor may they be redundantly implied on ASET or ASET1 cards. They may not appear as dependent degrees of freedom in other rigid elements or on MPC cards. Degrees of freedom declared to be independent by a rigid element can be made dependent by another rigid element or by an MPC card.

5. Rigid elements, unlike MPC's, are not selected through the Case Control Deck.

6. Forces of constraint are not recovered.

7. Rigid elements are ignored in heat transfer problems.

8. Nastran actually converts the CRBAR input card into the CRIGD3 card format, and thus processes a CRBAR card as if it were a CRIGD3 card. The following table shows the method of conversion, in free-field format:

```
CRBAR Card                    ===> Equivalent CRIGD3 Card
-----------------------------------------------------------------
CRBAR, EID, G1, G2, IC1, IC2, DC1, DC2
                              ===> CRIGD3, EID, G1, IC1, G2, IC2
                                   ,"MSET", G1, DC1, G2, DC2
```

9. See Section 1.4.2.2 for a discussion of rigid elements.

Input Data Card CRTRPLT Rigid Triangular Plate

Description: Defines a rigid triangular plate.

Format and Example:

1	2	3	4	5	6	7	8	9	10
CRTRPLT	EID	G1	G2	G3	IC1	IC2	IC3		abc
CRTRPLT	7	1	2	3	1236	3	3		ABC

+bc	DC1	DC2	DC3						
+BC									

Field	Contents
EID	Element identification number (Integer > 0)
Gi	Identification numbers of the triangular plate grid points. (Integers > 0)
ICi	Independent degrees of freedom in the global coordinate system for the element at grid points Gi (any of the digits 1-6 with no imbedded blanks. Integers ≥ 0 or blank.) See Remark 2.
DCi	Dependent degrees of freedom in the global coordinate system (any of the digits 1-6 with no imbedded blanks. Integers ≥ 0 or blank.) See Remarks 3 and 4.

Remarks:

1. Element identification number must be unique with respect to all other element identification numbers.

2. The total number of degrees of freedom specified for the reference grid points (IC1 IC2, and IC3) must be six; for example, IC1 = 1236, IC2 = 3, IC3 = 3. Further, they should together be capable of representing any general rigid body motion of the element.

3. If DC1, DC2, and DC3 are all zero or blank or if the continuation card is omitted, all of the degrees of freedom not in IC1, IC2, and IC3 will be made dependent.

4. The dependent (that is, constrained) degrees of freedom in a CRTRPLT element may not appear on OMIT, OMIT1, SPC, or SUPORT cards, nor may they be redundantly

implied on ASET or ASET1 cards. They may not appear as dependent degrees of freedom in other rigid elements or on MPC cards. Degrees of freedom declared to be independent by a rigid element can be made dependent by another rigid element or by an MPC card.

5. Rigid elements, unlike MPC's, are not selected through the Case Control Deck.

6. Forces of constraint are not recovered.

7. Rigid elements are ignored in heat transfer problems.

8. Nastran actually converts the CRTRPLT input card into the CRIGD3 card format, and thus processes a CRTRPLT card as if it were a CRIGD3 card. The following table shows the method of conversion, in free-field format:

```
CRTRPLT Card          ---> Equivalent CRIGD3 Card
--------------------------------------------------------------------
CRTRPLT, EID, G1,  G2, G3, IC1, IC2, IC3
,        DC1, DC2, DC3
                      ---> CRIGD3, EID, G1, IC1, G2, IC2, G3, IC3
                           ,"MSET", G1, DC1, G2, DC2, G3, DC3
```

9. See Section 1.4.2.2 for a discussion of rigid elements.

Input Data Card CRBE1 Rigid Body Element, Form 1

Description: Defines a rigid body connected to an arbitrary number of grid points.

Format and Example:

1	2	3	4	5	6	7	8	9	10
CRBE1	EID	IG1	IC1	IG2	IC2	IG3	IC3	╳	abc
CRBE1	103	11	1	12	2	13	4		ABC
+bc	╳	IG4	IC4	IG5	IC5	IG6	IC6	╳	def
+BC		14	35	15	6				CDF
+ef	"UM"	DG1	DC1	DG2	DC2	DG3	DC3	╳	ghi
+DF	UM	21	123	22	1	23	123456		EFI
+hi	╳	DG4	DC4	DG5	DC5	etc.	╳	╳	
+HI		24	456	25	2				

Field	Contents
EID	Element identification number (Integer > 0)
IGi	Identification numbers of the reference independent grid points (Integers > 0.)
ICi	Independent degrees of freedom in the global coordinate system for the preceding reference grid point (any of the digits 1-6 with no imbedded blanks. Integer > 0.) See Remarks 2, 3, and 5.
"UM"	BCD word that indicates the start of the data for dependent grid points.
DGi	Identification numbers of the dependent grid points (Integer > 0).
DCi	Dependent degrees of freedom in the global coordinate system for the preceding dependent grid point (any of the digits 1-6 with no imbedded blanks. Integer > 0.) See Remarks 4 and 5.

Remarks:

1. Element identification number must be unique with respect to all other element identification numbers.

2. The total number of degrees of freedom specified for the reference grid points (IC1 through IC6) must be six; for example, IC1=1, IC2=2, IC3=4, IC4=3,5, IC5=6. Further, they should together be capable of representing any general rigid body motion of the element.

3. The first continuation card is not required if less than four reference independent grid points are specified.

4. The dependent (that is, constrained) degrees of freedom in a CRBE1 element may not appear on OMIT, OMIT1, SPC, or SUPORT cards, nor may they be redundantly implied on ASET or ASET1 cards. They may not appear as dependent degrees of freedom in other rigid elements or on MPC cards. Degrees of freedom declared to be independent by a rigid element can be made dependent by another rigid element or by an MPC card.

5. A degree of freedom cannot be both independent and dependent for the same element. However, both independent and dependent components can exist at the same grid point.

6. Rigid elements, unlike MPC's, are not selected through the Case Control Deck.

7. Forces of constraint are not recovered.

8. Rigid elements are ignored in heat transfer problems.

9. Nastran actually converts the CRBE1 input card into the CRIGD3 card format by switching the "UM" BCD word to "MSET", and thus processes a CRBE1 card as if it were a CRIGD3 card.

```
CRBE1 Card             ---> Equivalent CRIGD3 Card
-----------------------------------------------------------------
CRBE1, EID, IG1, IC1, IG2, IC2, IG3, IC3
,    "UM", DG1, DC1, IG2, DC2, etc.
                       ---> CRIGD3,  EID, IG1, IC1, IG2, IC2, IG3, IC3
                              ,    "MSET", DG1, DC1, DG2, DC2, etc.
```

10. See Section 1.4.2.2 for a discussion of rigid elements.

Input Data Card CRBE2 Rigid Body Element, Form 2

Description: Defines a rigid body whose independent degrees of freedom are specified at a single grid point and whose dependent degrees of freedom are specified at an arbitrary number of grid points.

Format and Example:

1	2	3	4	5	6	7	8	9	10
CRBE2	EID	IG	C	G1	G2	G3	G4	G5	abc
CRBE2	9	8	12	10	12	14	15	16	ABC
+bc	G6	G7	G8	etc.					
+BC	20								

Field	Contents
EID	Element identification number (Integer > 0)
IG	Identification number of the reference grid point, to which all six independent degrees of freedom for the element are assigned (Integer > 0)
C	The dependent degrees of freedom in the global coordinate system for all the dependent grid points Gi (any of the digits 1-6 with no imbedded blanks. Integer > 0.) See Remark 2.
Gi	Identification numbers of the dependent grid points (Integers > 0)

Remarks:

1. Element identification number must be unique with respect to all other element identification numbers.

2. The dependent (that is constrained) degrees of freedom in a CRBE2 element may not appear on OMIT, OMIT1, SPC, or SUPORT cards, nor may they be redundantly implied on ASET or ASET1 cards. They may not appear as dependent degrees of freedom in other rigid elements or on MPC cards. Degrees of freedom declared to be independent by a rigid element can be made dependent by another rigid element or by an MPC card.

3. Rigid elements, unlike MPC's, are not selected through the Case Control Deck.

4. Forces of constraint are not recovered.

5. Rigid elements are ignored in heat transfer problems.

6. Nastran actually converts the CRBE2 input card into the CRIGD2 card format, and thus processes a CRBE2 card as if it were a CRIGD2 card. The following table shows the method of conversion, in free-field format:

```
CRBE2 Card             ---> Equivalent CRIGD2 Card
------------------------------------------------------------------
CRBE2, EID, IG, C, G1, G2, G3, etc.
                       ---> CRIGD2, EID, IG, G1, C, G2, C, G3, C, etc.
```

7. See Section 1.4.2.2 for a discussion of rigid elements.

Input Data Card CRBE3 Rigid Body Element, Form 3

Description: Defines the motion at a "reference" grid point as the weighted average of the motions at a set of other grid points.

Format and Example:

1	2	3	4	5	6	7	8	9	10
CRBE3	EID	╳	IG	IC	W1	C1	G1,1	G1,2	abc
CRBE3	14		100	1234	1.0	123	1	3	ABC
+bc	G1,3	W2	C2	G2,1	G2,2	G2,3	W3	C3	def
+BC	5	4.7	1	2	4	6	5.2	2	DEF
+ef	G3,1	G3,2	G3,3	W4	C4	G4,1	G4,2	G4,3	ghi
+EF	7	8		5.1	1	15	16		GHI
+hi	"UM"	DG1	DC1	DG2	DC2	DG3	DC3	╳	jkl
+HI	UM	100	14	5	3	7	2		JKL
+kl	╳	DG4	DC4	DG5	DC5	DG6	DC6	╳	
+KL									

Field	Contents
EID	Element identification number (Integer > 0)
IG	Reference grid point (Integer > 0)
IC	Global components of motion whose values will be computed at the reference grid point (any of the digits 1-6 with no imbedded blanks. Integer > 0)
Wi	Weighting factor for components of motion on the following card at grid points Gi,j (Real)
Ci	Global components of motion which have weighting factor Wi at grid points Gi,j (any of the digits 1-6 with no imbedded blanks. Integers > 0)

Gi,j	Grid point whose components Ci have weighting factor Wi in the averaging equations (Integers > 0)
"UM"	BCD word that indicates the start of the data for the components of motion at grid points DGi (Optional). The default is that all of the component in IC at the referent grid point IG, and no others, are included in the dependent component set (u_m)
DGi	Grid points with components DCi in (u_m) (Integers > 0)
DCi	Components of motion at grid point DGi (any of the digits 1-6 with no imbedded blanks, Integers > 0)

Remarks: 1. Element identification number must be unique with respect to all other element identification numbers.

2. Blank spaces may be left at the end of a Gi,j sequence.

3. The default for UM should be used except in cases where the user wishes to include some or all IC components in displacement sets exclusive from the (u_m) set. If the default is not used for UM:

   a. The total number of components in (u_m) (that is, the total number of dependent degrees of freedom defined by the element) must be equal to the number of components in IC (four in the above example).

   b. The components in UM must be a subset of the components mentioned in IC and (Gi,j; Ci).

   c. The coefficient matrix $[R_m]$ in the constraints equation $[R_m](u_m) + [R_n](u_n) = 0$ must be nonsingular.

4. The dependent (that is constrained) degrees of freedom in a CRBE3 element may not appear on OMIT, OMIT1, SPC, or SUPORT cards, nor may they be redundantly implied on ASET or ASET1 cards. They may not appear as dependent degrees of freedom in other rigid elements or on MPC cards. Degrees of freedom declared to be independent by a rigid element can be made dependent by another rigid element or by an MPC card.

5. Rigid elements, unlike MPC's, are not selected through the Case Control Deck.

6. Forces of constraint are not recovered.

7. Rigid elements are ignored in heat transfer problems.

8. Unlike the other rigid elements, this CRBE3 element and the CRSPLINE element cannot be converted into CRIGD2 or CRIGD3 elements. A Fortran subroutine (in single precision version and in double precision version) was written to handle these two special rigid elements.

Input Data Card CRSPLINE

Description: Defines multipoint constraints for the interpolation of displacements at grid points

Format and Example:

1	2	3	4	5	6	7	8	9	10
CRSPLINE	EID	D/L	G1	G2	C2	G3	C3	G4	abc
CRSPLINE	73	.05	27	28	123456	29		30	ABC
+bc	C4	G5	C5	G6	etc.				
+BC	123	75	123	71					

Field	Contents
EID	Element identification number (Integer > 0)
D/L	Ratio of the diameter of the elastic tube which the spline represents to the sum of the lengths of all segments. Default = 0.1 (Real > 0.)
Gi	Identification number of the ith grid point (Integer > 0)
Ci	Components to be constrained at the ith grid point (any of the digits 1-6 with no imbedded blanks, or blank) See Remark 3.

Remarks:

1. Element identification number must be unique with respect to all other element identification numbers.
2. Displacements are interpolated from the equations of an elastic beam passing through the grid points.
3. A blank entry in Ci indicates that all six degrees of freedom at Gi are independent. Since G1 must be independent, no field is provided for C1. Since the last grid point must also be independent, the last entry must be a Gi, not a Ci. For the example shown, G1, G3 and G6 are independent; G2 has six constrained degrees of freedom while G4 and G5 each have three.

4. The dependent (that is, constrained) degrees of freedom in a CRSPLINE element may not appear on OMIT, OMIT1, SPC, or SUPORT cards, nor may they be redundantly implied on ASET or ASET1 cards. They may not appear as dependent degrees of freedom in other rigid elements or on MPC cards. Degrees of freedom declared to be independent by a rigid element can be made dependent by another rigid element or by an MPC card.

5. Rigid elements, unlike MPC's, are not selected through the Case Control Deck.

6. Forces of constraint are not recovered.

7. Rigid elements are ignored in heat transfer problems.

8. Unlike the other rigid elements, this CRSPLINE element and the CRBE3 element cannot be converted into CRIGD2 or CRIGD3 elements. A Fortran subroutine (in single precision version and in double precision version) was written to handle these two special rigid elements.

USING PATRAN AND SUPERTAB AS PRE- AND POSTPROCESSORS TO COSMIC/NASTRAN

Robert R. Lipman

David Taylor Research Center
Numerical Structural Mechanics Branch (Code 1844)
Bethesda, Maryland 20084-5000

SUMMARY

Patran and Supertab are interactive computer graphics pre- and postprocessors that can be used to generate NASTRAN bulk data decks and to visualize results from a NASTRAN analysis. Both of the programs are in use at the Numerical Structural Mechanics Branch of the David Taylor Research Center (DTRC). This paper will discuss various aspects of Patran and Supertab including: geometry modeling, finite element mesh generation, bulk data deck creation, results translation and visualization, and the user interface. Some advantages and disadvantages of both programs will be pointed out.

INTRODUCTION

Interactive computer graphics is an integral part of finite element mesh generation and analysis results visualization. Gone are the days of typing GRID cards on a keypunch machine and pouring over endless pages of stress and displacement output. NASTRAN has plotting capabilities in a batch or interactive mode. However, in either mode, visual feedback while creating a finite element mesh is not possible and the results visualization capabilities are limited.

Presently, there are many finite element pre- and postprocessors that run on PC's, workstations, and mainframe computers. The pre- and postprocessors allow the user to interactively define geometry, approximate that geometry with a finite element mesh, apply loads and boundary conditions, create input for a finite element analysis program, and visualize results from the analysis. The programs provide a powerful, efficient, fast, and invaluable tool to an engineer to improve productivity.

Patran (ref. 1) and Supertab (refs. 2-5) are two of the more popular and widely used finite element pre- and postprocessors. Patran is a product developed and marketed by PDA Engineering of Costa Mesa, California. Supertab is a product developed and marketed by Structural Dynamics Research Corporation (SDRC) of Milford, Ohio. Both of these programs have interfaces to NASTRAN and can be used to generate finite element models and visualize analysis results. The scope of this paper covers the usage of Patran and Supertab as related only to COSMIC/NASTRAN (ref. 6) and not any other finite element analysis programs.

Several items about this paper should be noted. Both Patran and Supertab have a wide variety of features; I have not used, nor am I familiar with all of them. However, I have had extensive experience with Patran for the last 4 years and Supertab over the last year to generate finite element models of missile launchers, periscope masts and windows, propeller blades, and other Naval structures. In the Numerical Structural Mechanics Branch there are also several other experienced Patran and Supertab users. Any opinions expressed are my own and are not necessarily those of DTRC, the Navy, or the Department of Defense.

PATRAN OVERVIEW

The standard Patran software package consists of several integrated modules to generate geometry models consisting of curves, surfaces, and solids (P/Solid module); develop a finite element mesh consisting of nodes, elements, loads, boundary conditions, and material and physical properties (P/Fem); and visualize the geometry model, the finite element model (P/Image), and the analysis results (P/Post and P/Plot). Patran also has several optional modules that perform finite element analysis, mechanical dynamics, composite analysis, and thermal analysis. Interfaces to finite element analysis programs and IGES are also optional modules. Several utility programs are also provided, including a set of Fortran subroutines to access a Patran database directly without entering Patran.

Patran runs on many of the standard workstations, mainframe computers, and graphics devices. In the Numerical Structural Mechanics Branch, Patran version 2.2 is run on a network of Apollo workstations and Patran version 2.3 is run on a VAX with a Tektronix terminal. The Branch uses an interface to COSMIC/NASTRAN, but does not have any of the optional analysis modules.

Patran is a leased product. The lease fee is paid yearly and is determined by the number of concurrent users and the desired modules, interfaces, and graphics device drivers. The fee includes hotline support and software upgrades. Software upgrades are not released for all computers, modules, or interfaces at the same time. Presently, the Apollo version is one level behind the current VAX version.

SUPERTAB OVERVIEW

Supertab is one product of the I-DEAS (Integrated Design Engineering Analysis Software) family of software. The different families of software are: solid modeling (Geomod), engineering analysis (Supertab), system dynamics (Systan), drafting (Geodraw), and test data analysis (Tdas). Within each family there are different modules. The modules within Supertab are: pre/post processing, model solution, optimization, data loaders, and frame analysis. Most modules contain several tasks. Some of the tasks within the Supertab pre/postprocessing module are: geometry definition, mesh generation, model checking, and postprocessing. The data loader module of Supertab contains translators for all supported finite element analysis codes. I-DEAS software can be configured to contain only the required families and for some families, only the required modules. As part of the standard I-DEAS software package, a relational database management system (Pearl), an IGES translator, and several utility programs are provided, including software to integrate site-supplied software into I-DEAS as its own module.

Supertab also runs on many of the standard workstations, mainframe computers, and graphics devices. In the Numerical Structural Mechanics Branch, the I-DEAS product being used is called Supertab Plus version 4.0 running on a network of Apollo workstations. Supertab Plus consists of the pre/postprocessing and data loader modules of Supertab and the object modeling module of Geomod. In terms of geometry definition, the pre/postprocessing module has basic geometric modeling capablities, while the object modeling module has very powerful solid modeling capabilities.

Supertab is a licensed product. The user pays a one-time fee depending on the number of concurrent users and the desired software products. An optional yearly maintenance fee provides hotline support and software upgrades. As with Patran, software upgrades are not released for all computers, modules, or interfaces at the same time.

PATRAN GEOMETRY MODELING

Before a user creates a finite element mesh, the geometry model must first be generated. In Patran, the geometry model used to create a finite element model consists of points in space (Patran GRID entities), curves (LINE), surfaces (PATCH), and solids (HYPERPATCH). There are many ways to create these entities. For example, rotating a line about an axis will create a patch. A hyperpatch can be created from the linear interpolation of the region between two patches. The intersection between two patches creates a line. However, two hyperpatches cannot be intersected to create a third hyperpatch. Patches and hyperpatches could be reconstructed from the resulting lines of intersection between the individual patch faces of the original hyperpatches.

The mathematical formulation of the geometric entities is a parametric cubic. This representation has limitations; a line can go exactly through, at most, four grids; fitting a line through more than four grids will result in a least squares approximation. The approximation may be sufficient, or more than one line could be generated through the grids. The same problem occurs with generating a B-spline line. Given n grids, n-1 parametric cubic lines will be generated that represent a B-spline for those grids. Usually, the user would rather have one line through n grids. Having n-1 lines makes more entities to manipulate and keep track of.

Patches are always four-sided entities and hyperpatches are always six-sided entities. However, degenerate three-sided patches and degenerate five-sided hyperpatches are allowed. The sides of a patch are always single parametric cubic lines. A composite curve consisting of several lines defining the sides of a patch is not allowed. Therefore, the geometry model for a three-dimensional (3-D) object model will be divided into some combination of the geometric entities. The individual lines, patches, and hyperpatches defining the geometry model will be used to create the finite element model. Just as a finite element mesh normally is not discontinuous, the pattern of lines, patches, and hyperpatches should also not be discontinuous. Given the restrictions on the number of sides for patches (4 or 3) and hyperpatches (6 or 5), the desired finite element mesh, and the continuity of the geometric entities, an excessive number of geometric entities may be required to model some objects; and other objects will be almost impossible to model. This is more apparent when trying to divide an object into hyperpatches.

Patran has another solid modeling capability. This feature involves using boolean operations on solid primitives. The solid primitives available are bricks, cones, cylinders, elbows, spheres, and tori. A solid primitive can also be created from any collection of patches provided they form a closed surface. The boolean operations are difference, intersection, and union. The user can create a solid cube with a hole through the center by using a brick, a cylinder, and the difference operation. However, the primitives cannot be used directly to create a finite element mesh. First, the primitives have to be converted into geometric entities. For the cube with a hole, the desired geometric entity would be a set of hyperpatches. However, the resulting geometric entities are patches defining the original cube without a hole, patches defining the original cylinder, and the lines defining the intersection of the cube and cylinder. The hyperpatches can be constructed from those lines and patches. Using Patran primitives to generate geometric entities for finite element models is not very useful. Patran primitives are more useful in generating conceptual solid models of objects.

SUPERTAB GEOMETRY MODELING

Usually, before a finite element mesh is created in Supertab, two-dimensional (2-D) and 3-D regions, called mesh-areas and mesh-volumes, have to be defined. Mesh-areas and mesh-volumes are generated from curves and surfaces. There are two methods for creating curves and surfaces in Supertab. The first method is to generate a geometry model with the object modeling module of

Geomod and to transfer the curves and surfaces to Supertab. The second method is to use the geometry definition task in the pre/postprocessing module of Supertab.

The object modeling module of Geomod creates solid geometry models using 3-D primitives or 2-D cross-sections. The primitives availables are the same as in Patran. User-defined primitives can be generated by extruding or revolving cross-sections or by building an object from a set of cross-sections. The primitives can be cut, joined, or intersected using boolean operations. The curves and surfaces associated with the resulting primitives can be transferred to Supertab to be used in generating a finite element mesh.

The mathematical formulation of the curves and surfaces is a nonrational uniform B-spline (NURB). This representation allows one curve to be fit to any number of points. However, only planar outlines or cross-sections are allowed. Therefore, in Geomod and Supertab, a user-defined NURB must always lie in a plane. This is a limitation for some geometry models. For example, propeller blades are usually defined in terms of radial cross-sections. A radial cross-section is not allowed in Geomod or Supertab.

The geometry definition task of the pre/processing module of Supertab provides another alternative to create curves and surfaces that can be used to generate mesh-areas and mesh-volumes. This task is similar to object modeling in Geomod; however, only 2-D cross-sections can be created. Having only this geometry creation capability in Supertab is sufficient for many geometry models, making Geomod unnecessary.

PATRAN FINITE ELEMENT MODEL GENERATION

Nodes and Elements

In Patran, a finite element mesh is generated on the lines, patches, and hyperpatches that define the geometry model. For example, plate elements are generated on a patch. Two methods are available to create nodes and elements.

The first method is a two step process. In the first step, the GFEG command is used to create nodes on a line, patch, or hyperpatch. For a patch, the user specifies the number of nodes on each of two adjacent sides. This will create a mapped mesh of nodes from one side of the patch to the opposite side. The GFEG command allows for biasing the nodes and some limited mesh transitioning. The second step uses the CFEG command which specifies the type of element (for example: BAR, QUAD, HEX), the number of nodes per element, and a configuration code. The configuration codes are used to differentiate between different element types with the same number of nodes, such as CQUAD2 and CQUAD4. The number of elements generated depends on the pattern and number of nodes created with the GFEG command.

The second, and more powerful, method for generating a finite element mesh was implemented in the latest version of Patran. However, the method applies only to creating nodes and quadrilateral or triangular elements on patches. The MESH command is used to specify the type of element, the number of nodes per element, the configuration code, and the number of elements along all four sides of the patch or the approximate element edge length. Each side can have an arbitrary number of elements. This allows for whatever mesh transitioning or element size a user requires. A mesh smoothing command can be used to modify the resulting pattern of nodes and elements created by the MESH command.

The GFEG, CFEG, and MESH commands can create nodes and elements on more than one geometric entity at a time. For example, a geometry model consisting only of patches, might require that the GFEG command be repeated for each set of patches with the same pattern of nodes, and the

CFEG command used only once to generate the same type of elements on all the patches.

Material and Physical Properties, Loads, and Boundary Conditions

In Patran, material and physical properties, loads, and boundary conditions are applied to the finite element mesh with the PMAT, PFEG, and DFEG commands. Standard material models are available such as isotropic, orthotropic, and anisotropic. The physical properties specified are the same as would be required on a NASTRAN property card. Loads and boundary conditions can be applied to the nodes and elements associated with a geometric entity, a specific node or element, or all nodes or elements lying in a specified plane. Multi-point constraints (MPC's) can also be generated. Loads, boundary conditions, and physical properties can be defined by an algebraic function by using data entities or the FIELD command.

Equivalencing, Optimization, and Model Checking

When generating nodes on adjacent patches, nodes will be generated along the common boundary associated with each patch. Equivalencing eliminates one of the coincident nodes between adjacent geometric entities and readjusts the element connectivity. Optimization performs nodal resequencing based on based on bandwidth or wavefront. This capability is the same as the resequencing procedure in NASTRAN. The model checking capability checks the aspect ratio, warp, skew, taper, normals, and duplication of 2-D elements. If an element does not pass the check, the element can be split into two elements. No element checking is available for 3-D elements.

SUPERTAB FINITE ELEMENT MODEL GENERATION

Nodes and Elements

Supertab usually generates a finite element mesh on 2-D (mesh-areas) and 3-D (mesh-volumes) regions defined by curves and surfaces. The curves and surfaces come from the object modeling module of Geomod or the geometry definition task in the pre/postprocessing module of Supertab. A mesh-area is defined by a closed region of curves and a mesh-volume is defined by a closed volume of mesh-areas. A mesh-area can be defined by any number of curves and does not have to be planar. This is a very powerful tool to model any arbitrary 2-D region with only one mesh-area. The same is true for mesh-volumes. For a 3-D model only one mesh-volume, made up of multiple mesh-areas, is required.

Three methods for generating nodes and elements are available. The first method is mapped meshing, similar to the GFEG and CFEG commands in Patran. The number of elements along two adjacent "sides" of a mesh-area are specified. Because mesh-areas can have any arbitrary shape, mapped meshing is more appropriate for mesh-areas that are four "sided". Each "side" of a mesh-area, used for mapped-meshing, can be composed of any number of curves. The user specifies the number of elements along each curve of the two adjacent "sides" of the mesh-area. Biasing of the mesh is allowed. The element is specified by element type (rod, beam, plate, membrane, solid, etc.), element order (linear, parabolic, cubic), and element topology (triangle, quadrilateral, wedge, hexahedron, etc.). This specification does not distinguish between different NASTRAN elements that have the same element type, order, and topology, such as CQUAD2 and CQUAD4 elements.

Free mesh generation is the second method for generating nodes and elements. This capability can be used for mesh-areas and mesh-volumes and is similar to the Patran MESH command. To use free meshing, a global element size for a mesh-area or mesh-volume and the element type is specified.

Local element sizes and the number of elements per curve can also be specified. Free meshing is a very powerful capability, but care should be taken in creating mesh-areas and mesh-volumes and in specifying element sizes so that the resulting mesh is acceptable.

The third method for finite element mesh generation does not require any geometry model. Nodes can be created by entering or digitizing XYZ coordinates and copying, reflecting, or generating nodes between existing nodes. Rectangular, cylindrical, or spherical coordinate systems can be used. Elements can be created by picking the nodes for an element and copying, reflecting, extruding, or revolving existing elements. This method for finite element mesh generation can be used with nodes and elements created with either of the other two methods.

Material and Physical Properties, Loads, and Boundary Conditions

Material and physical properties are generated by creating tables of values for these properties. Only the properties that Supertab allows are permitted in the tables, which may not be sufficient to define all NASTRAN material and property cards. Both types of properties are associated with an element when the element is created. Loads and boundary conditions are applied to individual nodes and elements, nodes associated with elements, or nodes and elements on a geometric entity (curve, mesh-area, or mesh-volume). Load values can be defined by an algebraic function.

Equivalencing, Optimization, and Model Checking

Nodal equivalencing and resequencing in Supertab is similar to that in Patran. Nodes can also be resequenced by sweeping along a coordinate axis and sorting nodes based on nodal coordinates. Element checking is available for 2-D and 3-D elements. The adaptive meshing task in Supertab can be used to refine the finite element mesh based on element checking criteria.

BULK DATA DECK CREATION AND ANALYSIS RESULTS TRANSLATION

Patran

The ultimate goal of any finite element pre- and postprocessor is to create input for an analysis program and to tranlate results for the postprocessor. The programs used by Patran to accomplish this are PATCOS (PATran to COSmic/nastran translator) and COSPAT (COSmic/nastran to PATran translator). PATCOS and COSPAT (refs. 7-9) are developed and supported by PDA Engineering. Although PDA is currently updating the translators, the current versions of PATCOS and COSPAT have several bugs and have not been updated to include many bulk data cards that are new or which were missing from previous versions. Fortunately, when the Numerical Structural Mechanics Branch originally obtained COSPAT and PATCOS, PDA supplied the Fortran source code, which allowed us to bring COSPAT and PATCOS up-to-date by implementing many bug fixes, additions, and enhancements.

To create a bulk data deck, a Patran neutral file must first be created. The neutral file is an ASCII file containing all geometric and finite element model information and is generated by Patran. PATCOS reads the neutral file and generates a bulk data deck. If a new type of bulk data card is required, then PATCOS has to be modified. It is also possible to generate elements in Patran that are not supported by NASTRAN or PATCOS, such as a 15-noded wedge. The user has to be aware of the capabilities of PATCOS when generating a finite element model.

COSPAT serves two functions. The first is to read in a NASTRAN bulk data deck and generate a Patran neutral file. If COSPAT does not recognize a particular NASTRAN card type, then nothing is written to the neutral file. The resulting neutral file can be read into Patran. Once in Patran, the finite element model can be displayed and used for postprocessing.

The other function of COSPAT is to translate NASTRAN results into a format that can be read into Patran. COSPAT reads in displacement and stress data blocks that are written to a NASTRAN UT1 file with an OUTPUT2 statement. COSPAT generates up to three different Patran results files. One file contains nodal translations and rotations. The second file contains element centroidal stresses. The third file is generated only if NASTRAN computes nodal stresses (CIHEXi, CQUAD2, CTRIA2, etc.). Any of these three files can be used in Patran to visualize analysis results. With a user-written postprocessor, any type of data can be written in the Patran results files format so the data can be visualized with Patran.

Supertab

To generate a bulk data deck in Supertab, the finite element model must first be written to an I-DEAS Pearl database. The Pearl database is read by a program that generates the bulk data deck. Using the Pearl database to create the bulk data deck is a time-consuming procedure. A more efficient way to generate a bulk data deck might be to create it directly from the model file or from an I-DEAS universal file. A universal file is an ASCII file containing the geometry model, finite element model, analysis results, and viewing parameters. A universal file is similar to a Patran neutral file. Because Supertab cannot differentiate between a CQUAD1, CQUAD2, and CQUAD4 element, all linear quadrilateral thin shell elements will be translated to a CQUAD2 element. This problem also affects other element types. The user has to edit the bulk data deck to change elements to the desired element type. The source code for the program which generates the bulk data deck is not available.

To read in results from a NASTRAN analysis the data loader module of Supertab is used. Similar to PATRAN, the NASTRAN data loader reads data blocks that are written to a NASTRAN UT1 file with an OUTPUT2 statement. In addition to displacement and stress data blocks, data blocks for strains, forces, strain energy, and eight others which define the finite element model are required: CSTM, GPL, GPDT, EPT, MPT, GEOM2, GEOM3, and GEOM4. CSTM is generated only when a coordinate system definition card is included in the bulk data deck. If the default coordinate system is being used, then a dummy coordinate card must be included in the bulk data deck to force the generation of the data block CSTM. Although the user might not be interested in a particular type of output (for example, strain energy), the data block for that type of output is still required.

The NASTRAN data loader creates an I-DEAS universal file which can be read into Supertab. The analysis results can then be used for postprocessing. The data loader cannot read a bulk data deck to create a universal file of the finite element model. The source code for the data loaders is available from SDRC.

VISUALIZATION

Both Patran and Supertab have similar capabilities for visualizing the geometry model, finite element model, and analysis results. There are an infinite number of ways to display either type of model. The user has control over: viewing angles, which parts of the model are to be displayed, the color assigned to different entities (curves, mesh-areas, patches, element types, etc.), how to draw an entity (shrink elements, a circle or dot for a node, etc.), entity labels, display options, etc. The display option can be wireframe, hidden line, continuous tone (Supertab) or fill-hide (Patran), or shaded image. For graphics terminals and workstations with hardware 3-D rotations and shading, the user should be able to dynamically rotate a model drawn with any display option. Patran cannot display a shaded image of a finite element model. Supertab has an advanced display capability known as ray tracing. Ray traced images can have shadows, reflections, and transparency. However, ray tracing, as implemented in Supertab, is extremely computationally intensive (several days for one image on an Apollo DN580-T).

To visualize analysis results, several types of display options are available. They include: deformed geometry, animation of modal vibrations, contour plots, color fringe plots, vector plots, fill-hide plots, beam shear and moment diagrams, and XY graphs. The appropriate types of display options can be dynamically rotated. Various attributes of these types of displays can be set by the user. Patran can assign colors to elements based on analysis results or a value such as element or material ID. Currently, COSPAT does not generate Patran beam results files that can be used for beam shear and moment diagrams in Patran.

USER INTERFACE, DOCUMENTATION, AND BUGS

Patran

The Patran user interface is a mixture of a command-driven and menu-driven input system. The user interface can be used in a command line mode or on-screen menu mode. In the command line mode, the user enters whatever commands are desired. However, to do some tasks, a menu pick is required. To pick a particular menu item, the user enters the number associated with it. Some tasks can be executed by entering a command or using menu picks with the same results. Commands are also available to jump to particular menus. In the on-screen menu mode, menu items are chosen by using the cursor (controlled by a mouse, thumbwheels, etc.) to pick from a dynamic menu. Some commands still have to be entered in the on-screen menu mode. Other commands can be set by using the cursor. A less ambiguous and more structured user interface would be desirable for Patran.

Most commands in Patran can be divided into several catagories: commands for creating geometric entities (GR, LI, PA, HP), commands for creating the finite element model (GFEG, CFEG, PFEG, DFEG, etc.), and commands prefaced by SET, SHOW, or RUN. The SET and SHOW commands are used to set and show the value of almost 300 parameters. Generally, only a small subset of the parameters might have to be set. For example, labels can be turned on by entering SET,LABEL,ON. Currently, there are over 50 RUN procedures. The RUN procedures allow the user to do such things as generate hidden line plots, assign colors to elements, or compute contour line values.

The text that is entered for many of the commands, SET/SHOW parameters, and RUN procedures is not obvious. If the user did not know how to set the number of line segments plotted per parametric cubic line, entering SET,NLSPPC,10 would not be obvious. Therefore, documentation is essential. Patran documentation is divided into major tasks such as: creating geometric entities, creating a finite element model, visualizing models and results, and using SET, SHOW, and RUN commands. Generally, a description in words and graphics is given alphabetically for each command in a task. There are also functional listings of SET, SHOW, and RUN commands. For some types of commands there are conceptual descriptions of what can be done, along with related commands that might be used. The documentation is a complete reference of any capability or option in Patran. When running Patran, a user may access on-line help consisting of command descriptions.

No program of Patran's size is without bugs. PDA publishes a technical bulletin every month or two that lists a few known bugs and possible work-arounds. The release notes for a new version of Patran contain a list of bugs that have been fixed. The user does not have a list of all known bugs.

Supertab

Supertab's user interface is a tree-structured menu-driven system. The user picks a menu response which results in: (1) another menu, (2) a required alphanumeric response, or (3) a system action (for example, an element is generated). Menu items can be picked by using the cursor or by typing in the command. More than one command can be entered at a time, allowing the user to move through several menus at one time. Menu picks that require an alphanumeric response (file names, coordinate values, various parameters, etc.), generally have a default value which is used if the user hits the return key. In addition to the current menu, the user can pick from a global menu that has commands that can be executed anytime within a module.

Generally, every task in Supertab has its own main menu with many submenus. The menu system can be confusing. If the user knows what he wants to do, it is not always obvious what the command name might be or under what menu to find the command. As part of the standard documentation, menu guides are provided which list all the commands for each menu in a hierarchical form. The standard manuals do not have an explanation of all of the commands. Rather, the manuals introduce the use of Supertab conceptually and through step by step examples. This is a good method; however, the manuals are almost useless if a user is trying to determine what a specific command does. Optional reference manuals are available that give a description of each command. When running Supertab, a user may access on-line help consisting of command descriptions, a command search capability, and some overviews and methods.

As with Patran, Supertab is not without bugs. SDRC publishes a quarterly update of all known bugs and work-arounds and hints, limitations, and extended documentation for some features. Although the list of bugs is extensive, it is not complete because not all bugs are reported to SDRC.

OTHER FEATURES

Patran

When Patran is run, a session file is generated containing everything that was entered at the keyboard. This file can be used to reconstruct the model or to model objects with similar shape but different dimensions. For example, a session file that made hyperpatches defining a cylinder could be edited to change the radius and length. The session file could be rerun to create a new model with new dimensions. A session file can only be input at the beginning of Patran. If a file of commands is to be entered while already in Patran, the "read file" option under the geometry menu can be used.

Macros can be defined which create a user-defined text string that will represent several commands. A file (OPTION.SET) is executed everytime a new Patran database is opened. This file can be used to configure the user's working environment and to define macros. A replay file can be generated in Patran that contains all the graphics that appear on the terminal. This file can be replayed later with a utility program. A hardcopy file can generated and processed with another utility program that sends the graphics to a plotter (Calcomp, laser printer, etc.).

A named component is an entity that is a user-defined collection of geometric and/or finite element model entities. This provides a simple method to refer to a large number of entities of different types. New named components can be created by mirroring, rotating, scaling, or translating existing named components.

Supertab

When Supertab is run, a program file can be generated. Similar to a Patran session file, a program file contains everything that was entered at the keyboard. A program file can be input to Supertab at any time. In addition to Supertab commands, the program file can contain Fortran-like statements including arithmetic operators, mathematical functions, character strings, and variables, as well as commands to extract data from Supertab. Program files in this language can be written for any application and executed from Supertab. The program file can contain "read" and "write" statements to prompt the user for input, in the same way Supertab prompts for input.

Macros can be defined as they are defined in Patran. A user-defined program file (USERPROF.PRG) is executed everytime a new module is entered in I-DEAS. A picture file can be generated of a graphic image and replayed within Supertab or with a utility program and sent to a plotter.

Supertab has an adaptive meshing capability which, given the finite element analysis results, will refine the finite element mesh based on selected criteria. For example, a region of a mesh with high stress gradients could be refined to have a higher mesh density in that region.

CONCLUSIONS

Either Patran or Supertab can be used successfully as a pre- and postprocessor to COSMIC/NASTRAN. Each program has advantages and disadvantages. The user will have to decide which program is better depending on his finite element pre- and postprocessing needs. My personal opinion is that Supertab and Geomod are superior to Patran. However, if only simple finite element models are required, Patran might be a better program to use once the user has become familiar with Patran. Both programs need a convenient capability for fitting a surface through a specified set of points.

Patran

The solid primitive capability in Patran would be more useful if the primitives could be used directly for finite element mesh generation. The finite element model generation capabilities are simple and straightforward. The MESH command should be extended to handle generating solid elements on hyperpatches. The bulk data deck generation and results translation processes are fast and efficient provided the user has access to the source code for PATCOS and COSPAT. PDA Engineering should provide up-to-date versions of PATCOS and COSPAT so that the user does not have to become involved with the source code. The user interface leaves a lot to be desired. A new user will find it difficult to come up to speed to generate even a moderately complex finite element model.

Supertab

The solid modeling capabilities of Geomod are very powerful. However, the restriction that cross-sections be planar is a limitation. The finite element mesh generation capabilities of Supertab are also very powerful; however, not all element types can be generated. Creating an I-DEAS Pearl database slows the bulk data deck creation process. The user should not have to generate NASTRAN data blocks that define the finite element model to do postprocessing of analysis results. The user interface is very good; however, a user can get lost in the tree-structured menu-driven input system. The user interface allows a new user to come up to speed very quickly to generate complex geometry and finite element models.

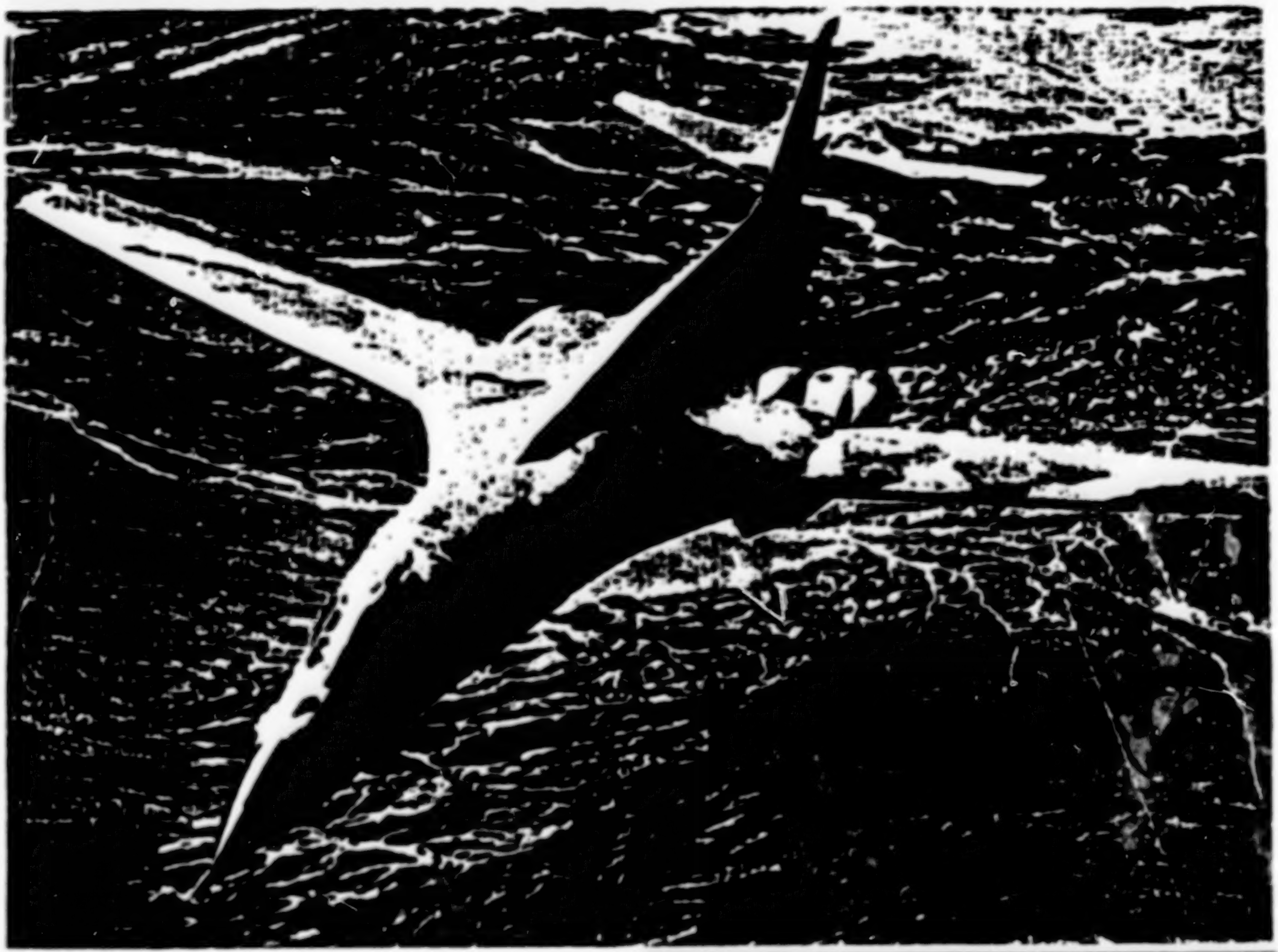
REFERENCES

1. "PATRAN Plus User Manual," PDA Engineering, Costa Mesa, CA, 1987.
2. "Getting Started With I-DEAS," SDRC, Milford, OH, 1988.
3. "I-DEAS User's Guide," SDRC, Milford, OH, 1988.
4. "I-DEAS Geomod User's Guide," SDRC, Milford, OH, 1988.
5. "I-DEAS Supertab User's Guide," SDRC, Milford, OH, 1988.
6. "NASTRAN User's Manual," NASA SP-222(06), Washington, DC, 1983.
7. "PAT/COSMIC-NASTRAN Application Interface Guide," PDA Engineering, Costa Mesa, CA, 1984.
8. Bender, L.C., and M.P. Johnson, "Enhancing Your COSMIC NASTRAN Usage with PATRAN," *Sixteenth NASTRAN User's Colloquium,* NASA CP-2505, National Aeronautics and Space Administration, Washington, D.C., pp. 31-38, April 1988.
9. Libby, D.H., "COSMIC/NASTRAN - PATRAN Interface," *Thirteenth NASTRAN User's Colloquium,* NASA CP-2373, National Aeronautics and Space Administration, Washington, D.C., pp. 133-151, May 1985.

INCREASING MARKETABILITY AND PROFITABILITY OF PRODUCT LINE THRU
PATRAN & NASTRAN

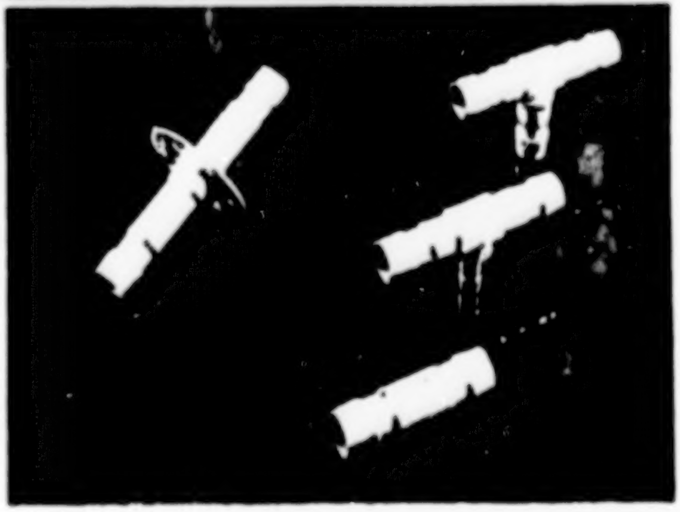
ART HYATT
DEUTSCH METAL COMPONENT

DEUTSCH
METAL COMPONENTS
14800 S. Figueroa, Gardena, CA 90248 (213) 323-6200

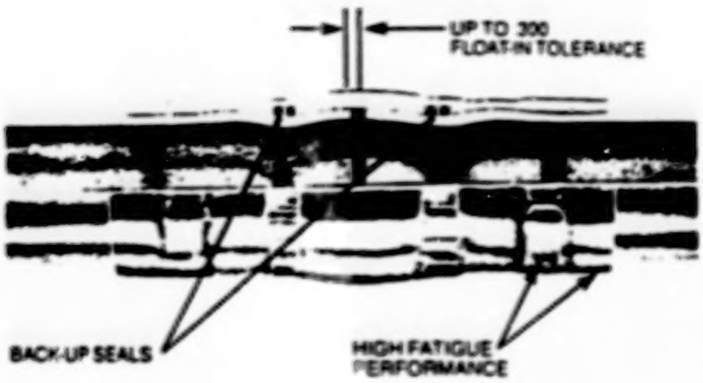


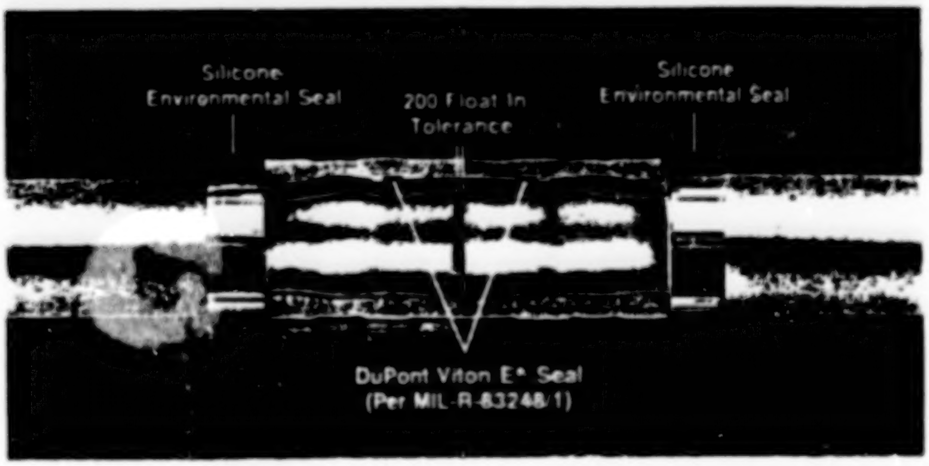
INTRODUCTION

Deutsch Metal Components Manufactures advanced Fluid System Components for the Aeropsace, Marine, and Petrochemical Industries. These fittings permanently connect Pipe and/or Tubing Systems and are recognized under the trade names, **Permaswage® and Pyplok®.**



The above photo shows the B-1B Bomber. This is one of the many types of aircraft that uses the Deutsch fittings.


UP TO 300 FLOAT-IN TOLERANCE
BACK-UP SEALS
HIGH FATIGUE PERFORMANCE


Silicone Environmental Seal
200 Float In Tolerance
Silicone Environmental Seal
DuPont Viton E* Seal (Per MIL-R-83248/1)

FITTING DESIGN

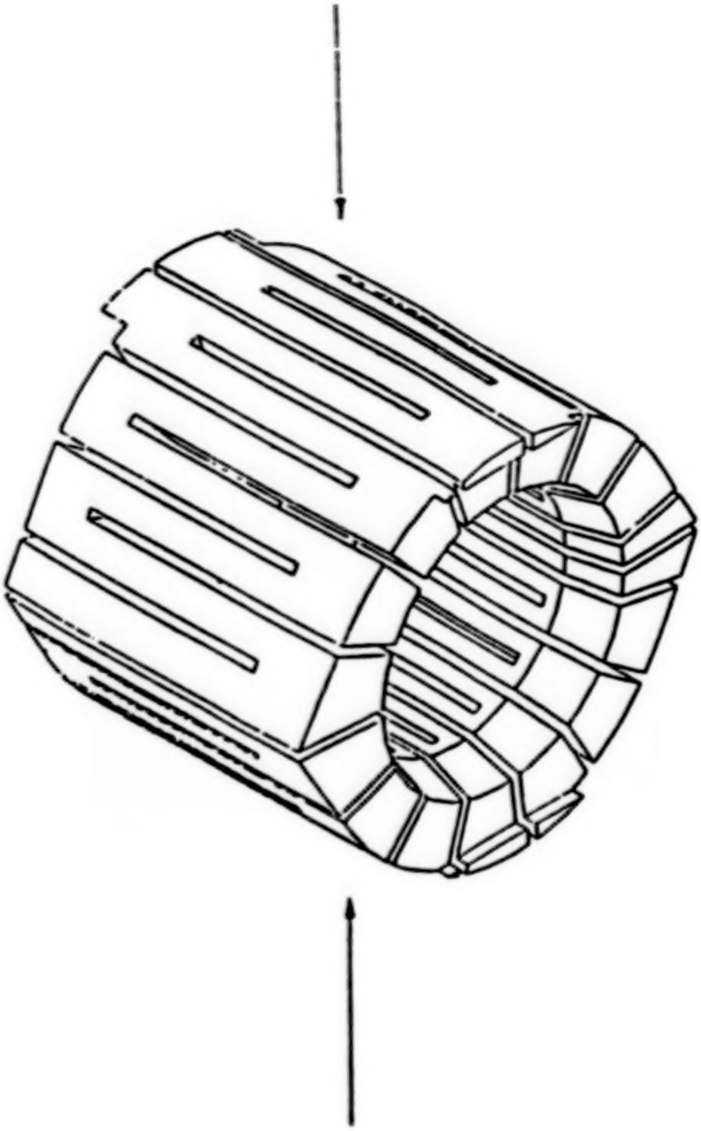
The upper photo shows a cut-a-way Permaswage® fitting connecting together two pieces of tubing. The fitting is slid into the tubes and then swaged or crimped resulting in a permanent light weight connection. Completed connections feature 2 seals on each side of the fitting. One is a positive metal-metal seal the other is a silicone back-up seal.

The lower photo shows the Pyplok® fitting and pipe connecting system which is similar in concept to Permaswage®, but is used on piping systems in the Ship Building/Ship Repair and Petrochemical Industries. Both systems are available in the standard Pipe/Tube fitting configuration, such as; Coupling, Tees, Elbows, Reducers, etc.

SWAGE TOOL

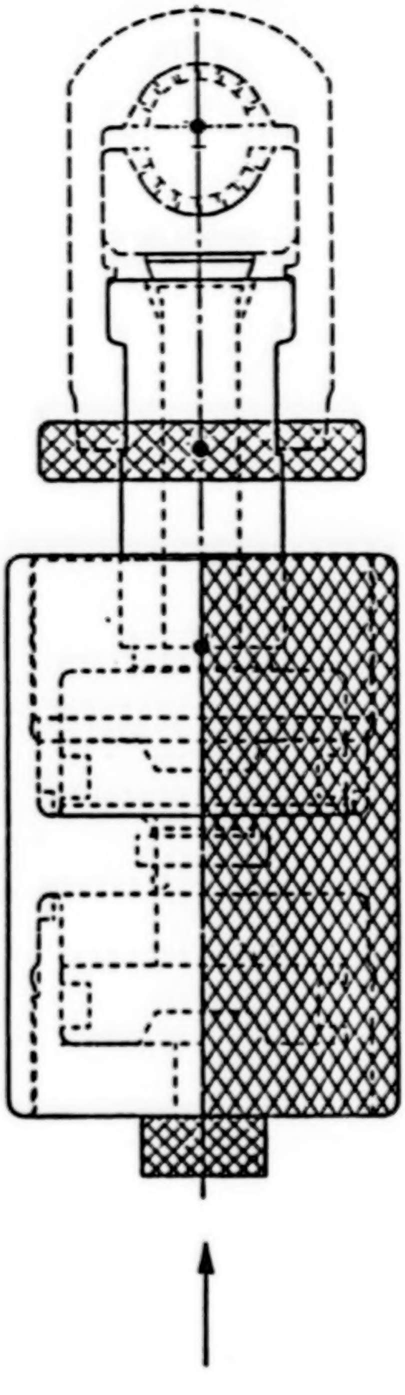
This Swaging Tool is one of the models of the product line which is the **subject** of the paper. There are 6 different models in the product line: Model 5, 10, 20, 30, 40, and Model 55. For simplicity in this paper only one model will be discussed. The swaging tool which performs the swaging or crimping of these fittings is hydraulically actuated.

The photo above shows this tool in operation. The upper left hand slide shows the top portion of the tool which contains a slotted Die which transforms axial or linear force from an axial direction to radially acting force. The next slide shows a detail of this Swage Die.



SWAGING DIE

his is a 3 "D" view of the slotted die. This is a ather unique design. It transforms axial force to radial force. The fitting is placed inside for Swaging.

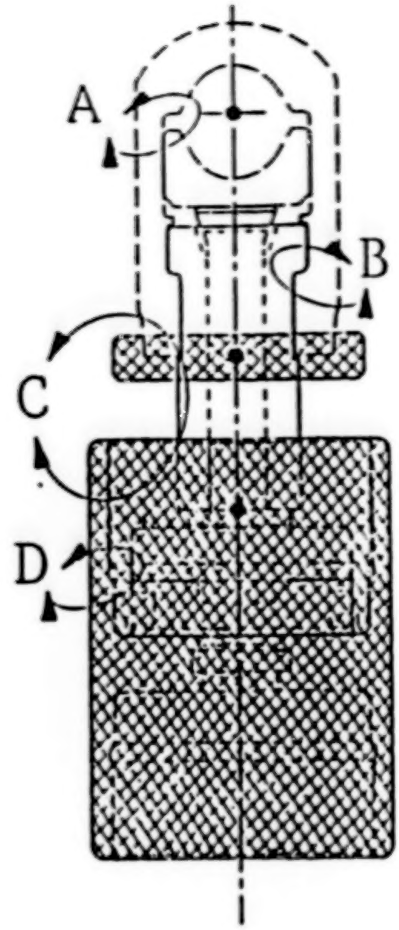


EXISTING SWAGE TOOL

This slide shows the previous generation Tool. Pressurized oil enters at the bottom of the Tool and actuates the upper and lower pistons. These pistons force the lower die block against the slotted die which performs the swaging operation.

DESIGN OBJECTIVE:

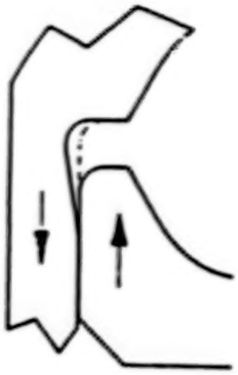
- **To double the operational cycle life of the product.**
- **MCD felt that with some redesign of the tool, cycle life could be substantially extended.**



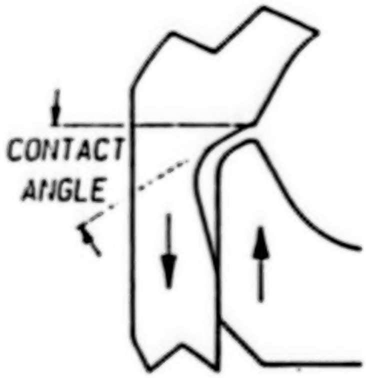
APPROACH USED TO ACCOMPLISH DESIGN OBJECTIVE

- First, we submitted the existing Swage Tool to finite element analysis.
- Second, we determined the stress level required. This was based on a S-N Curve for the selected material and the operational cycle life requirement.
- Third, we concentrated our efforts on the highly stressed areas noted above.
- The first area analyzed was "A", the force on this area is 1/2 the total tonnage of the tool. Model 55 tool would have a total force of 110,000# of which 55,000# would be concentrated under worst conditions on this area.
- "B" is the reaction area.
- "C" reacts the forces trhu the Strut and Top Cap back into the cylinder.
- "D" is the area where the Top Cap Threads mate the cylinder.
- These areas will be discussed in detail starting with "A".

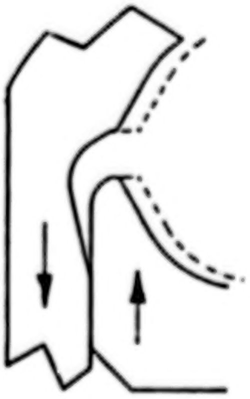
AREA "A"



1st the radius was increased as shown above. This reduced the stress.

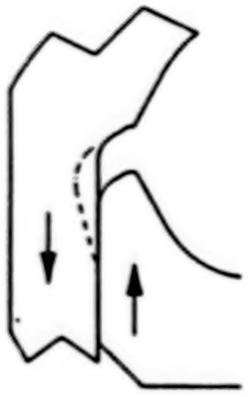


2nd the contact angle which was 0°, was changed. This reduced the stress some more. It should be noted at this point, different angles were tried from 0°-45°. The optimim angle was found to be 30°.

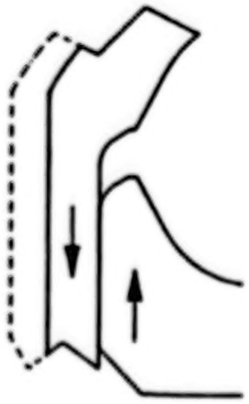


3rd the compression area was then made smaller. Again this reduced the moment and the associated stress.

AREA "A" CONT.

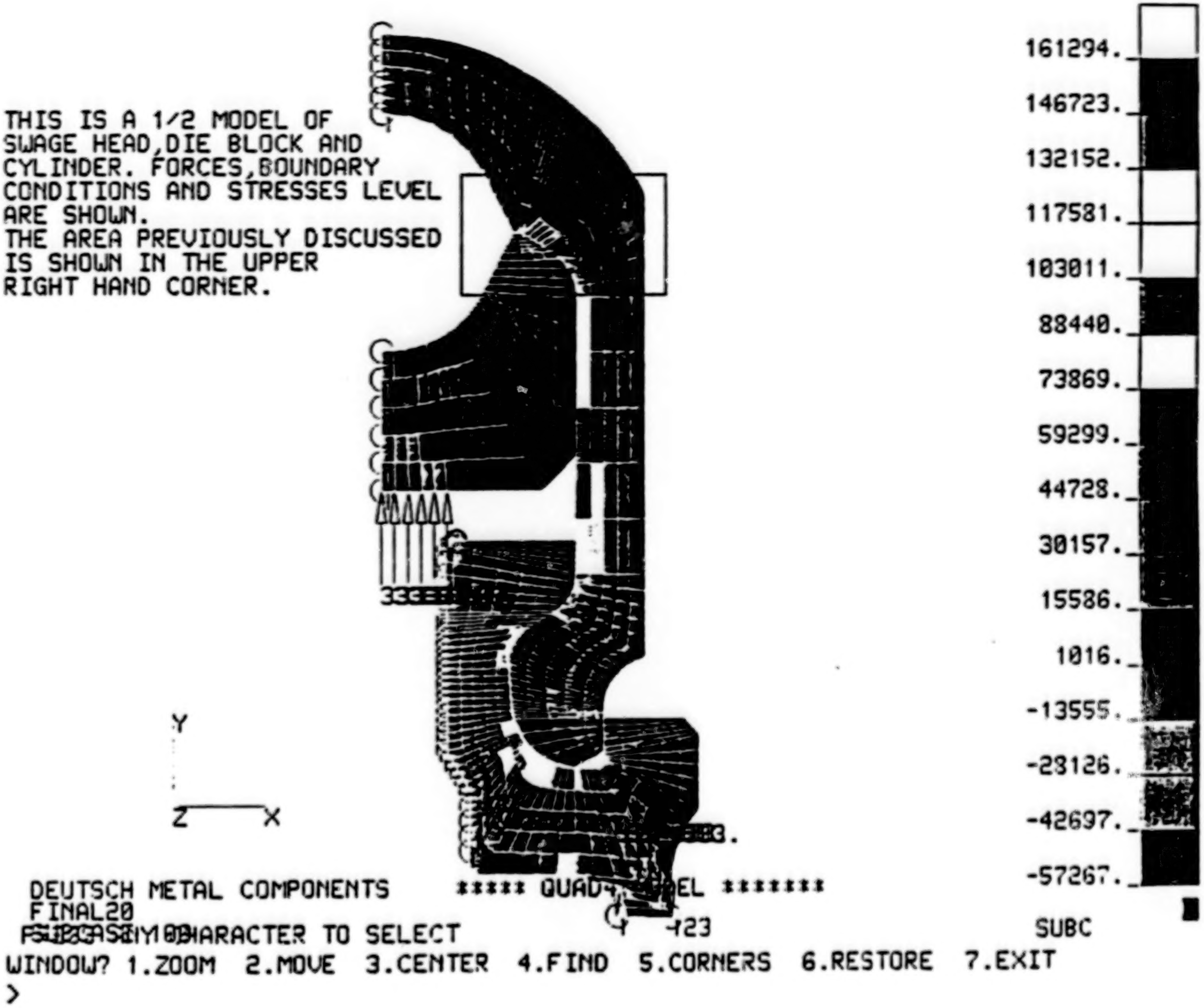


4th the contact area was reduced and the radius moved inward. This too lowered the stress level because the area across the high stress section was increased. The contact stress was performed by hand calculations.



5th the overall size was reduced. This was possible because the stress level had been substantially reduced in all areas.

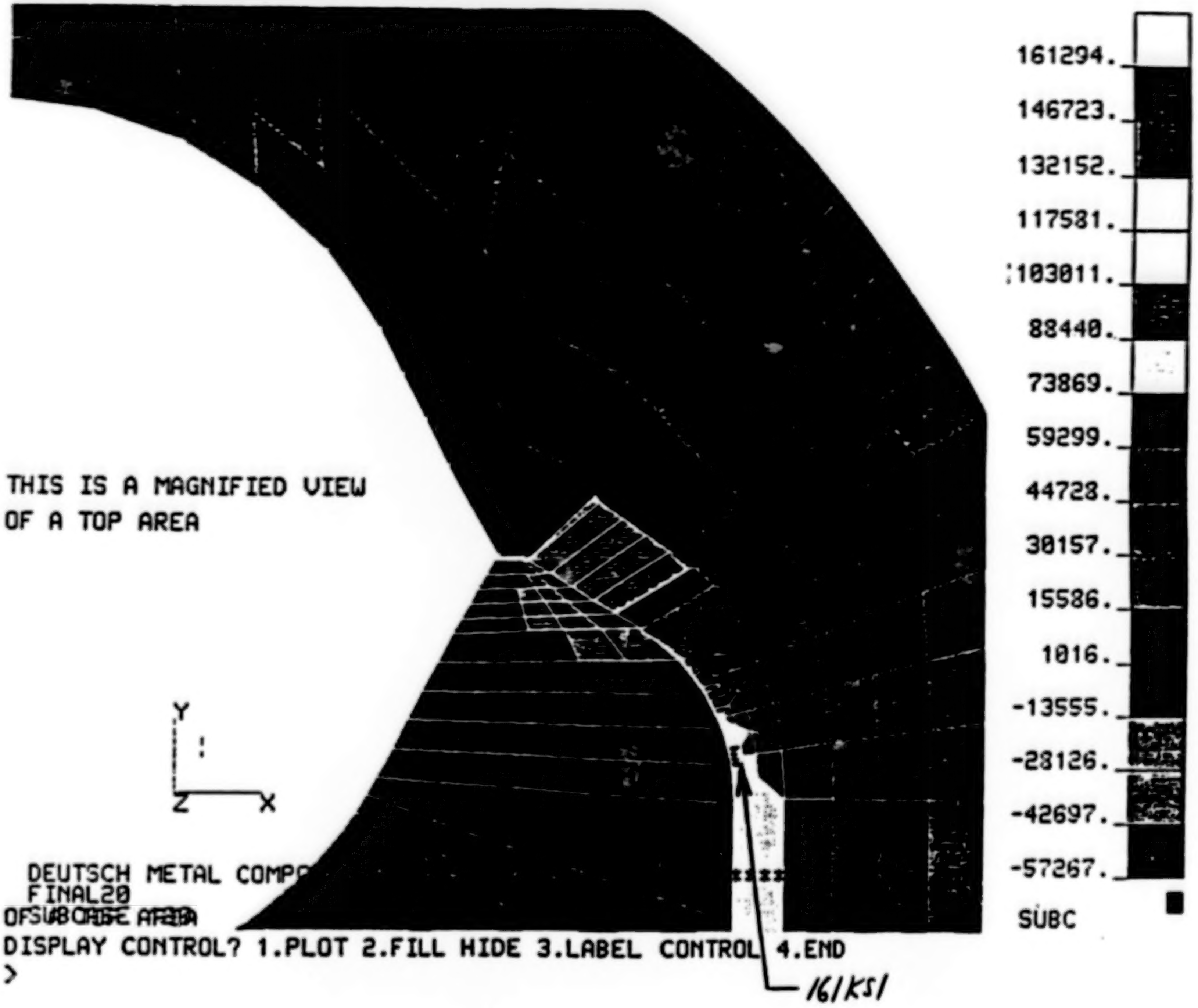
The actual final FEA are shown on the following slides.



THIS IS A 1/2 MODEL OF
SWAGE HEAD,DIE BLOCK AND
CYLINDER. FORCES,BOUNDARY
CONDITIONS AND STRESSES LEVEL
ARE SHOWN.
THE AREA PREVIOUSLY DISCUSSED
IS SHOWN IN THE UPPER
RIGHT HAND CORNER.

161294.
146723.
132152.
117581.
103011.
88440.
73869.
59299.
44728.
30157.
15586.
1016.
-13555.
-28126.
-42697.
-57267.
Y
Z
X
3.
423
SUBC

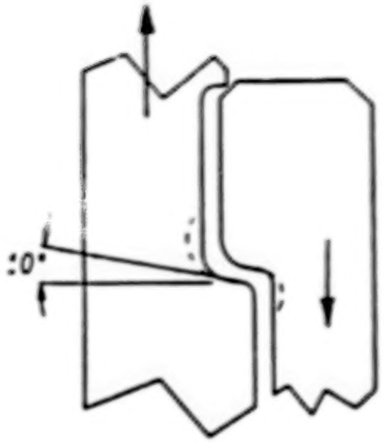
```
DEUTSCH METAL COMPONENTS    ***** QUAD4 MODEL *******
FINAL20
FSUBCASE1Y10CHARACTER TO SELECT
WINDOW? 1.ZOOM  2.MOVE  3.CENTER  4.FIND  5.CORNERS  6.RESTORE  7.EXIT
>
```



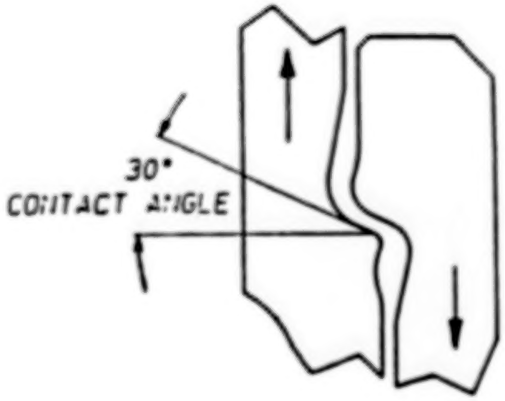
```
THIS IS A MAGNIFIED VIEW
OF A TOP AREA

 DEUTSCH METAL COMPA
 FINAL20
OFSUBCASE AREA
DISPLAY CONTROL? 1.PLOT 2.FILL HIDE 3.LABEL CONTROL 4.END
>
```

161294.
146723.
132152.
117581.
103011.
88440.
73869.
59299.
44728.
30157.
15586.
1016.
-13555.
-28126.
-42697.
-57267.
SUBC
Y
Z
X
161KSI

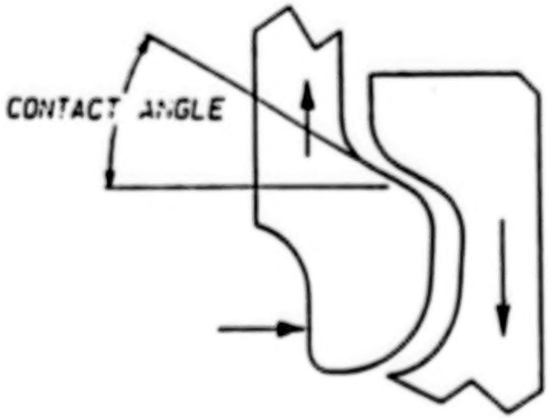
THE SECOND AREA ANALYZED WAS THE REACTION PRESSURE AREA AT "B"



1st the radius was increased as shown above. This reduced the stress.

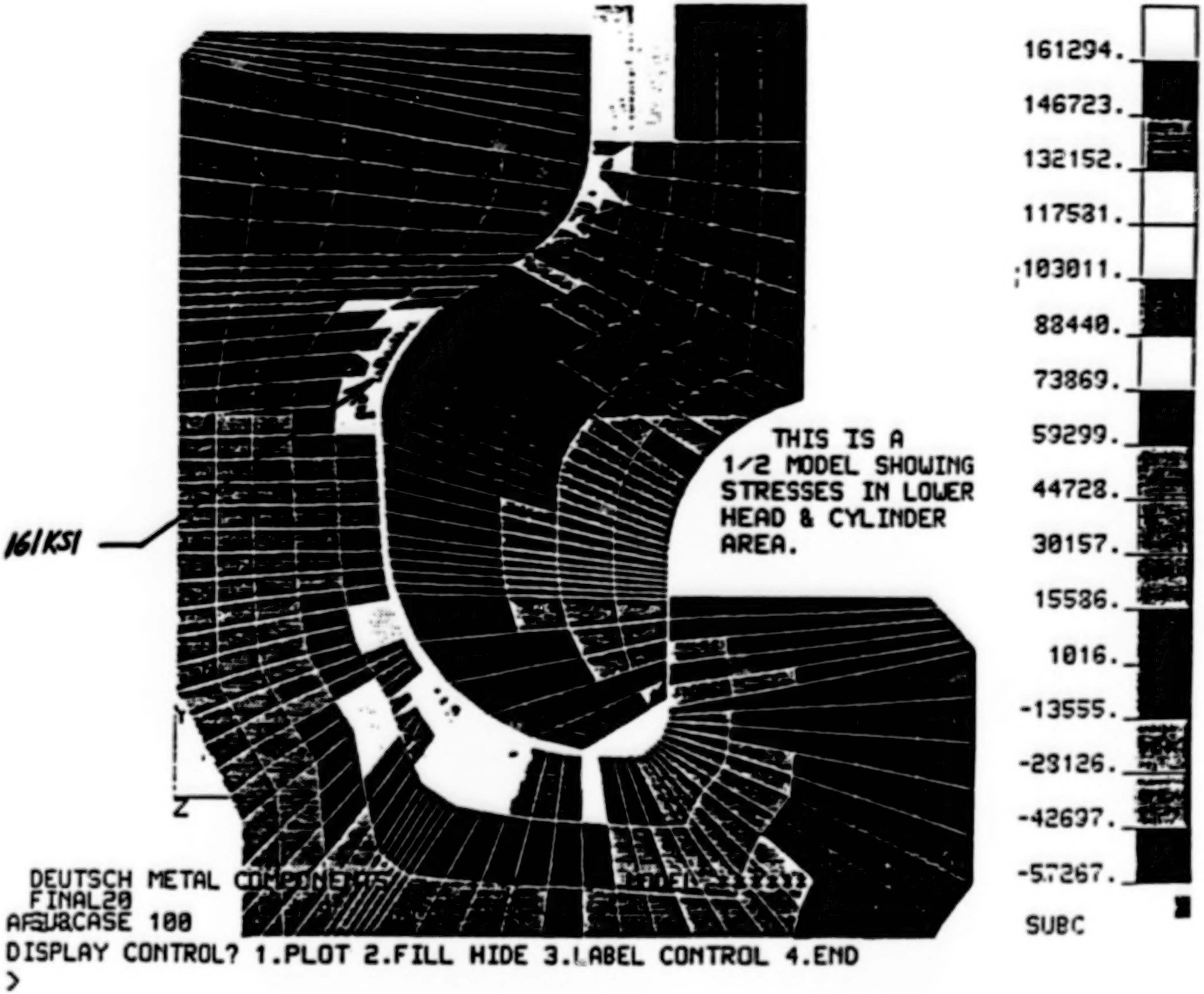


2nd the contact angle which was 10°, was changed to the optimum angle of 30°. The contact area was also changed again. Both of these changes lowered the stresses.



3rd The radii on both parts were made more generous & blended together. This again reduced the stress not only on the head, but also the cylinder. The result was a smaller and lighter cylinder and head. Actual FEA slides follow.

BEST COPY AVAILABLE

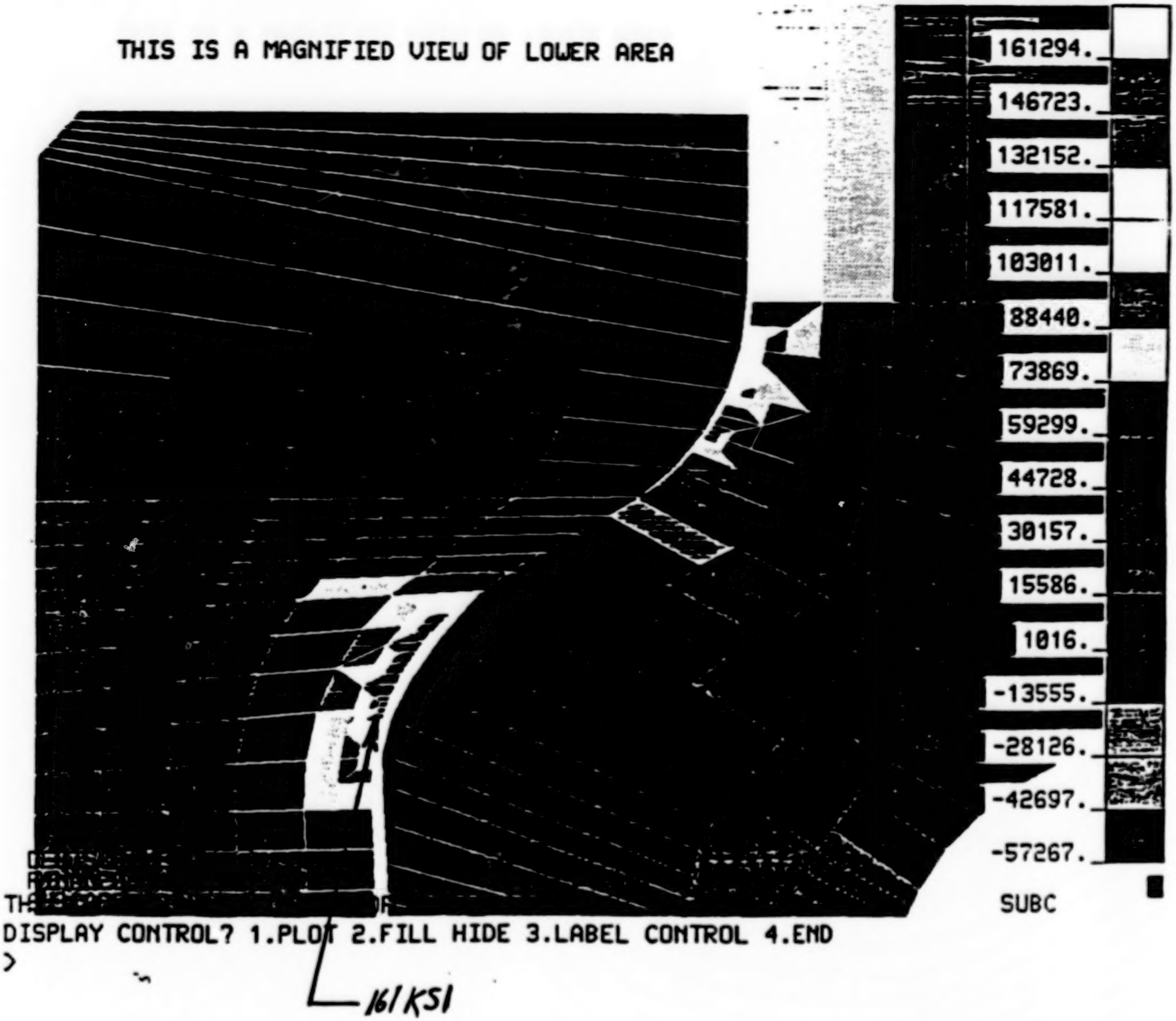

161294.
146723.
132152.
117581.
103011.
88440.
73869.
59299.
44728.
30157.
15586.
1016.
-13555.
-28126.
-42697.
-57267.
SUBC
Z

161KSI

THIS IS A
1/2 MODEL SHOWING
STRESSES IN LOWER
HEAD & CYLINDER
AREA.

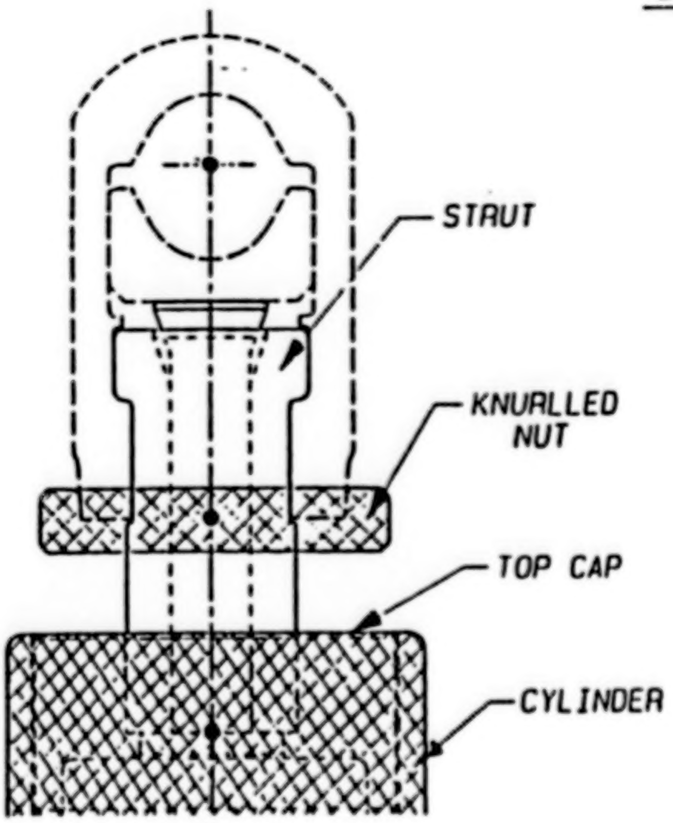
```
DEUTSCH METAL COMPONENTS
FINAL20
AFSUBCASE 100
DISPLAY CONTROL? 1.PLOT 2.FILL HIDE 3.LABEL CONTROL 4.END
>
```

BLANK PAGE

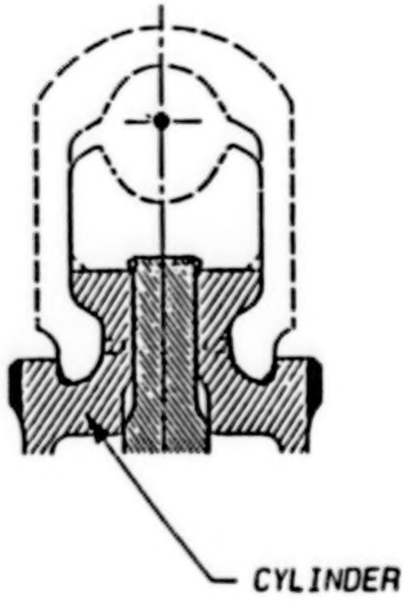

THIS IS A MAGNIFIED VIEW OF LOWER AREA
161294.
146723.
132152.
117581.
103011.
88440.
73869.
59299.
44728.
30157.
15586.
1016.
-13555.
-28126.
-42697.
-57267.
SUBC
161 KSI

```
DISPLAY CONTROL? 1.PLOT 2.FILL HIDE 3.LABEL CONTROL 4.END
>
```

4 PIECE DESIGN

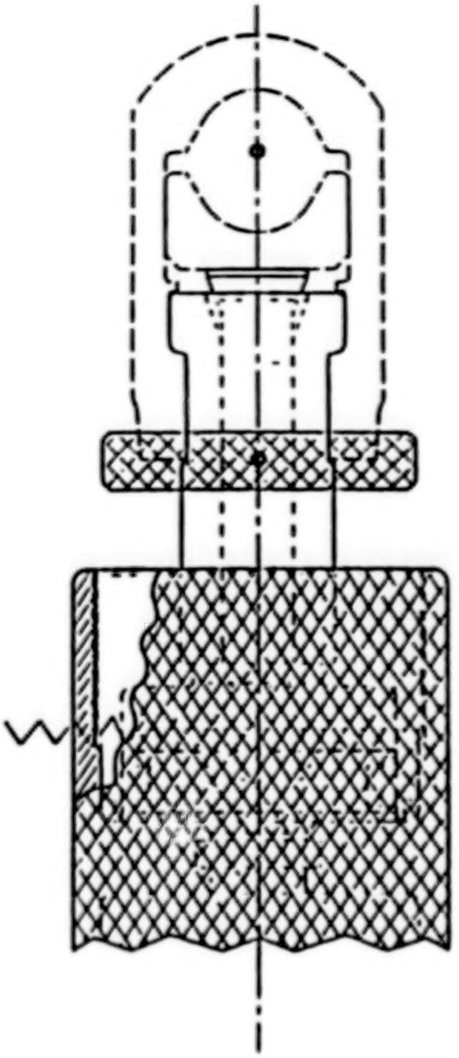


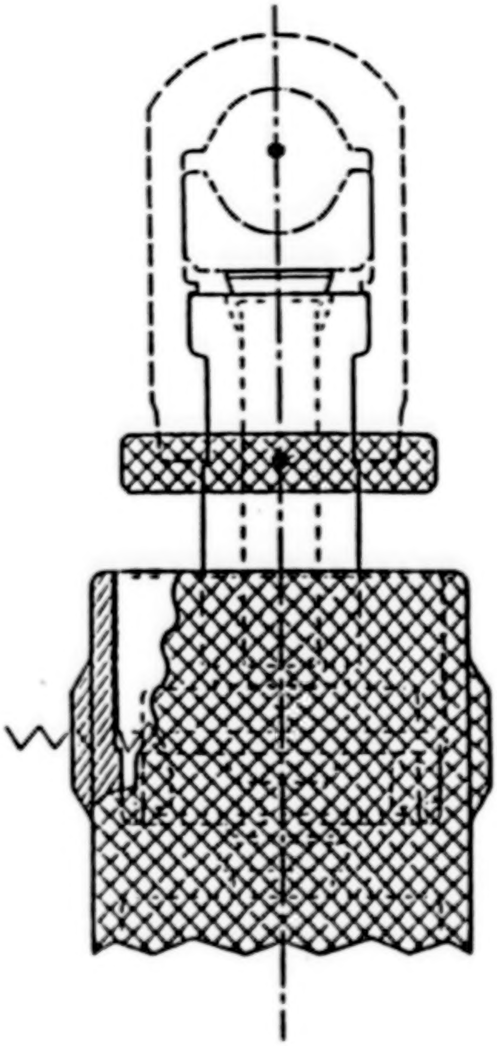
1 PIECE DESIGN



THE 3RD AREA ANALYZED WAS "C"

Knowing the forces involved it was an easy step to go from a (4) piece design, to a (1) piece design. The Strut, Knurled Nut, Top Cap and Cylinder were all made part of the Cylinder. This completely eliminated 3 parts. Finite elements again checked all high stress areas. The first analysis on the new design found several areas which had excessively high stress. Adding more material to the O.D. of the Cylinder and changing to a larger radius as shown resulted in acceptable stress levels.

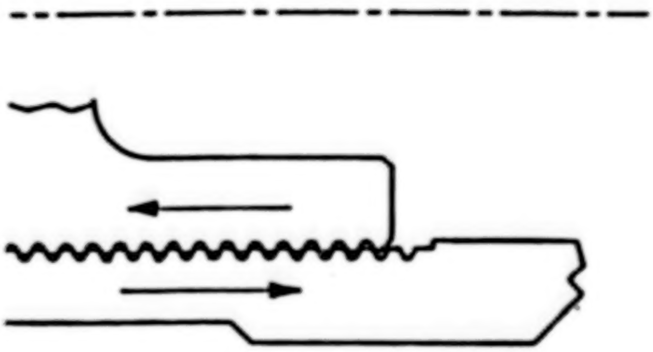




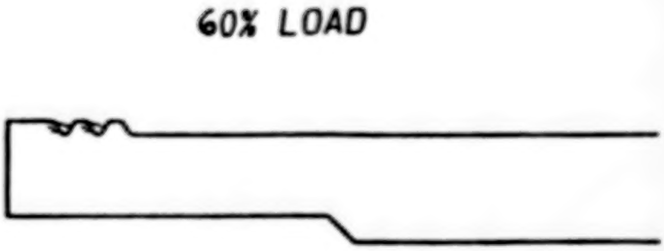
THE 4TH AREA ANALYZED WAS "D"

Lab tests showed that premature failure occurred at the 1st and 2nd thread area before we could reach the required cycle life. To solve the problem we made the following engineering modifications:

FIRST — More material was added to the O.D. of the cylinder where the failure occurred. Repeated tests showed very little improvement in the cycle life.



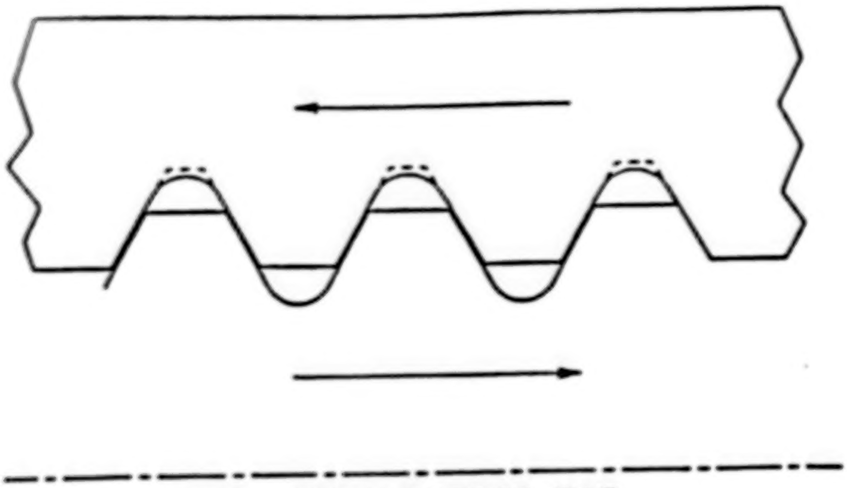
ACTUAL THD. LENGTH



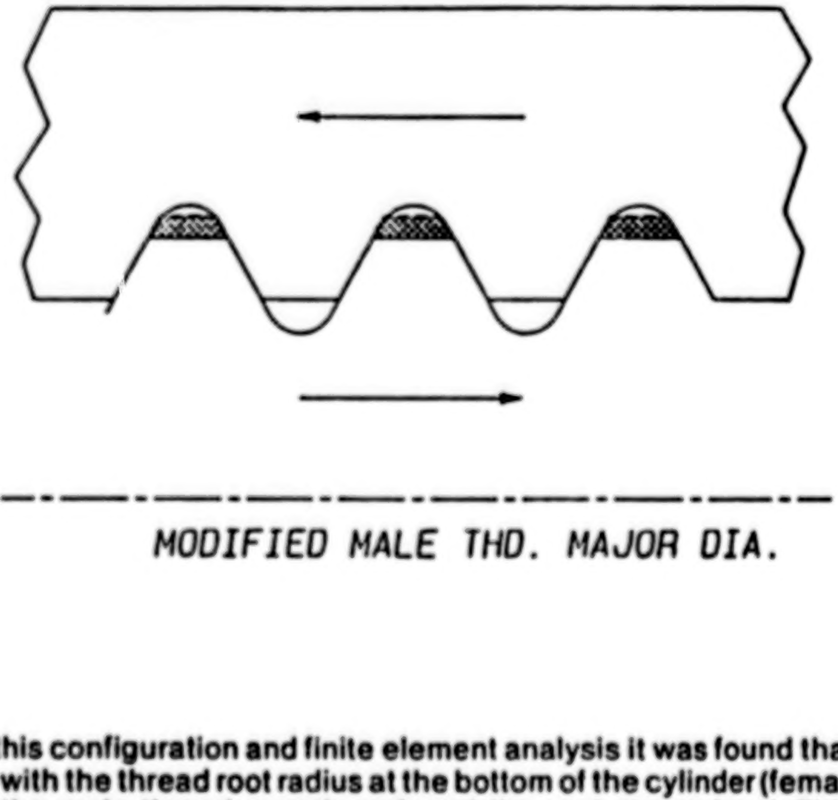
THD LENGTH FOR FEA

SECOND — The thread area was submitted to finite element analysis. The analysis was made for only 2 threads and was in 2 "D". 30% of the load was applied to each thread. This loading was applied because studies have shown that 60% of the load on a screw thread is concentrated on the first 2 threads.

FEA showed the weak area up immediately. It was, as expected, at the root of the thread.

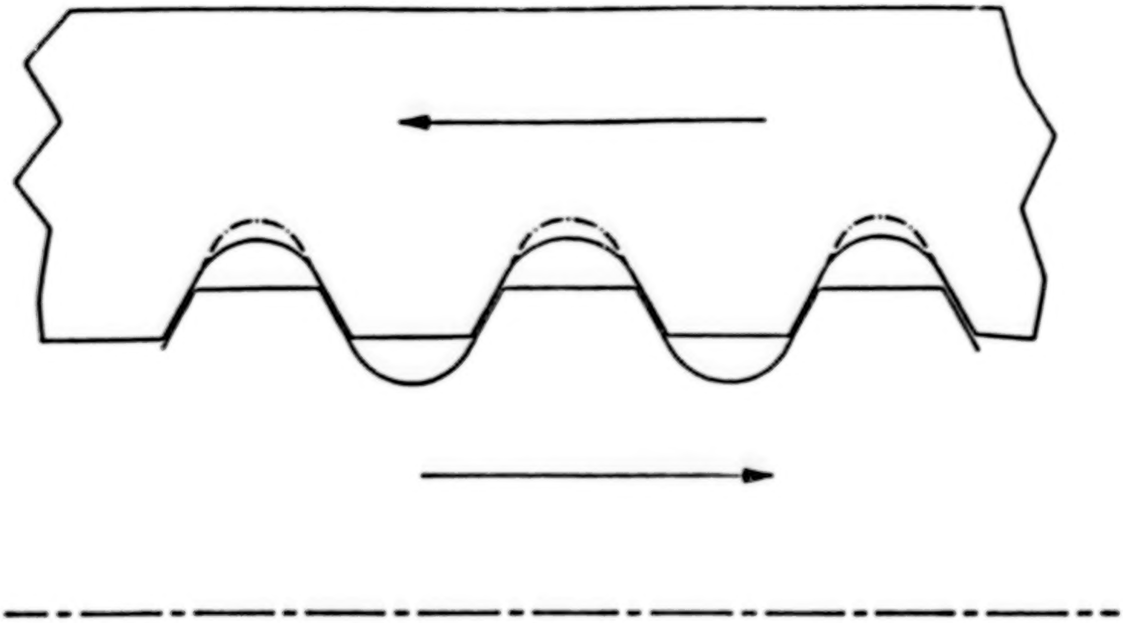


MOD. FEMALE ROOT RAD.



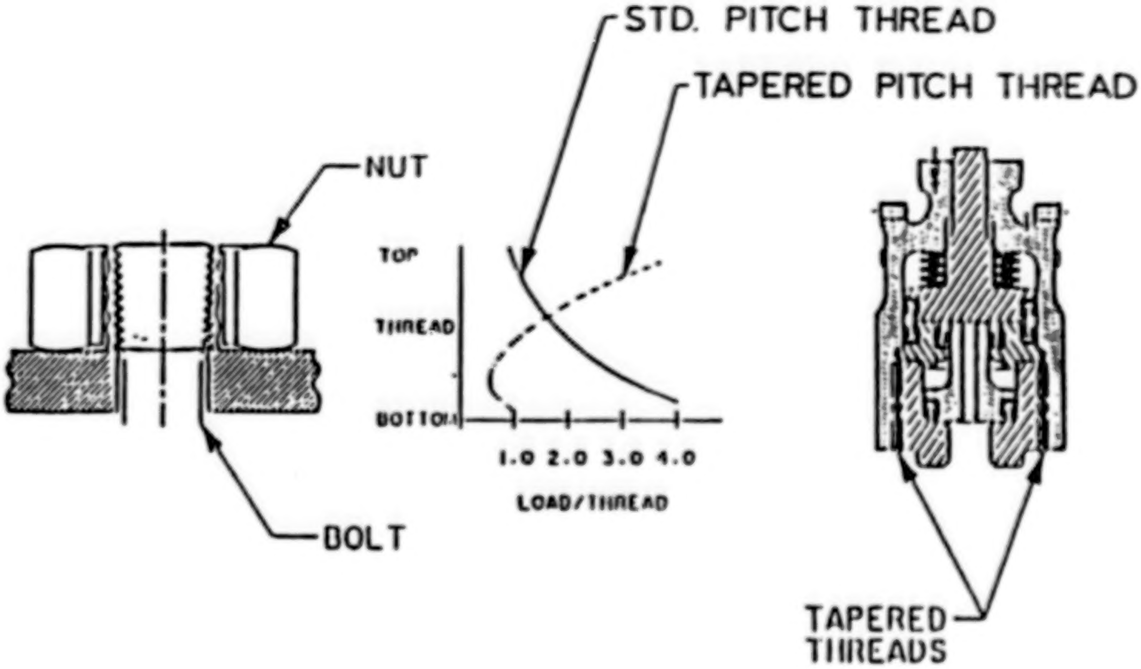
MODIFIED MALE THD. MAJOR DIA.

THIRD — Using this configuration and finite element analysis it was found that a modified UNJ thread with the thread root radius at the bottom of the cylinder (female) thread, as well as, on the male thread, greatly reduced the root stress level. This now became a modified thread because the standard UNJ thread only specified that the male have a root radius. With this female root radius it was also necessary to modify the O.D. of the male thread to prevent interference. Prototype models were then made, however, we still did not meet our required 100,000 cycle life requirement.

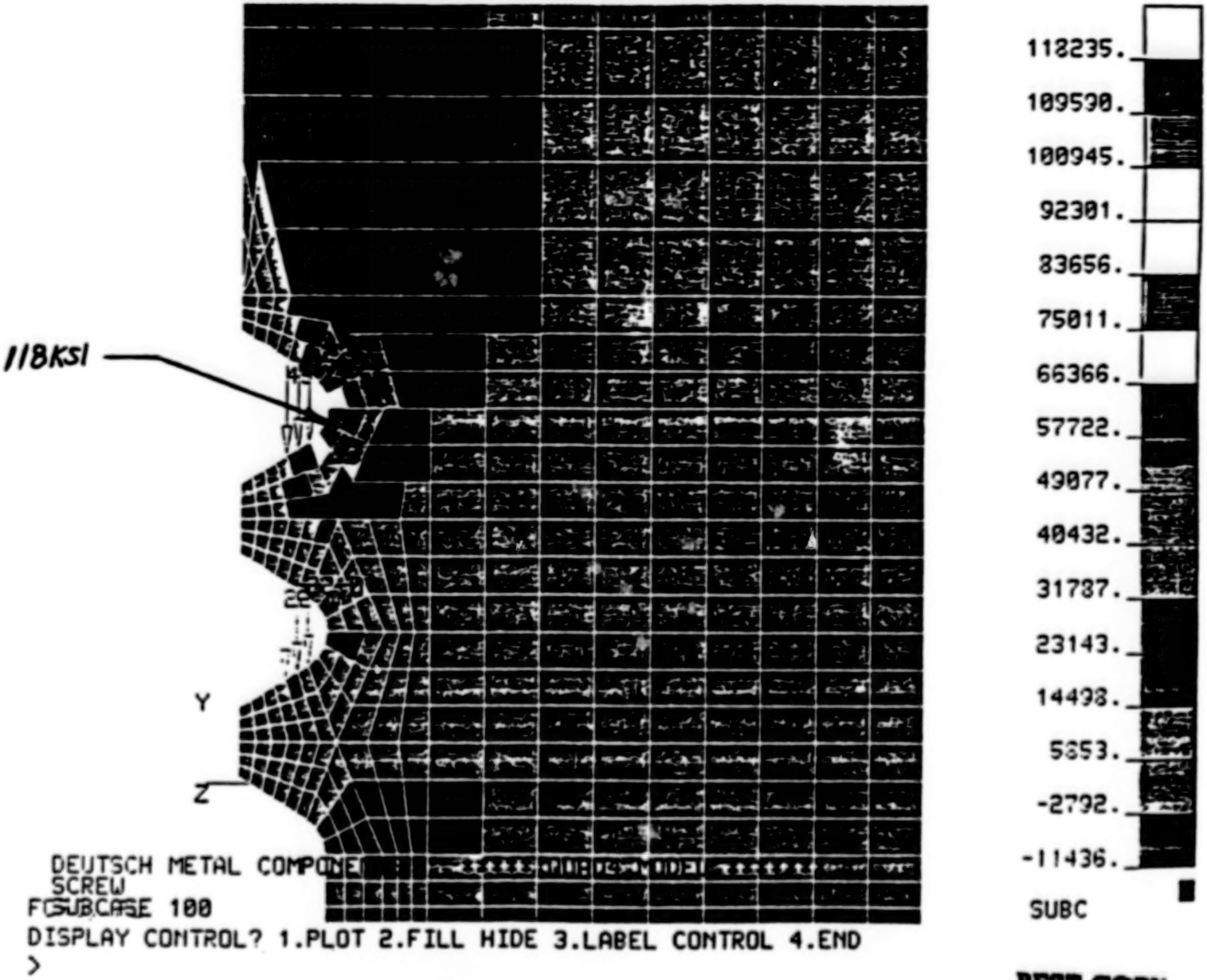


LARGER THAN STD. UNJ RAD.
WITH SMALL CONTACT AREA

FOURTH — The next step was to return to FEA which showed that an increase in the root radius over and above the recommended UNJ radius would again lower stress levels. This was then tested and it was proven to add significantly to the cycle life. This solved the tool cycle life problem, however, because the area of contact was reduced the design could be a problem in manufacturing. If manufacturing made the slightest deviation in thread configuration failure could occur in the shear mode.

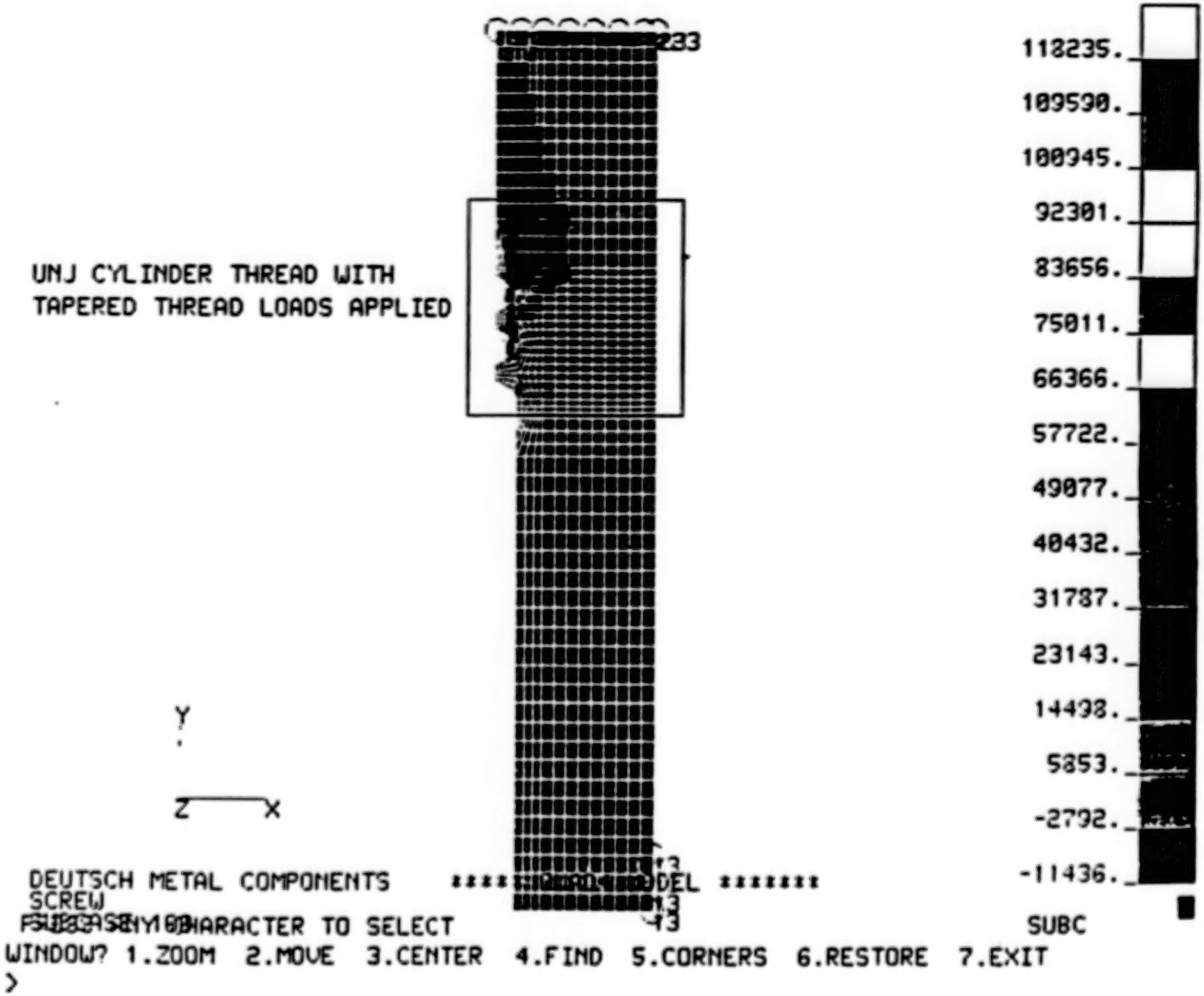

STD. PITCH THREAD
TAPERED PITCH THREAD
NUT
TOP
THREAD
BOTTOM
1.0 2.0 3.0 4.0
LOAD/THREAD
BOLT
TAPERED THREADS

FIFTH — **TAPERED PITCH DIAMETER** As a final step, we found an article written for ASME on, "Effect of taper on screw-thread load-distribution". This article basically states that a .006 inch/inch taper of the pitch dia would result in a fairly distributed thread load. This design was first programmed on the CAD System and then subjected to finite element analysis. Results looked outstanding. Models were made and subject to test. The 100,000 cycle requirement was not only met but continuous cycle testing went well beyond the 100,000 cycles without any signs of failure in this area. It should be noted at this point that the cylinder wall was also reduced. The amount of reduction was almost 50% less than the original concept. This whole exercise proves that adding material does not necessarily make for a stronger product. The following slides show the final FEA results.


118KSI
118235.
109590.
100945.
92301.
83656.
75011.
66366.
57722.
49077.
40432.
31787.
23143.
14498.
5853.
-2792.
-11436.
SUBC
Y
Z

```
DEUTSCH METAL COMPONE[illegible] MODEL
SCREW
F SUBCASE 100
DISPLAY CONTROL? 1.PLOT 2.FILL HIDE 3.LABEL CONTROL 4.END
>
```

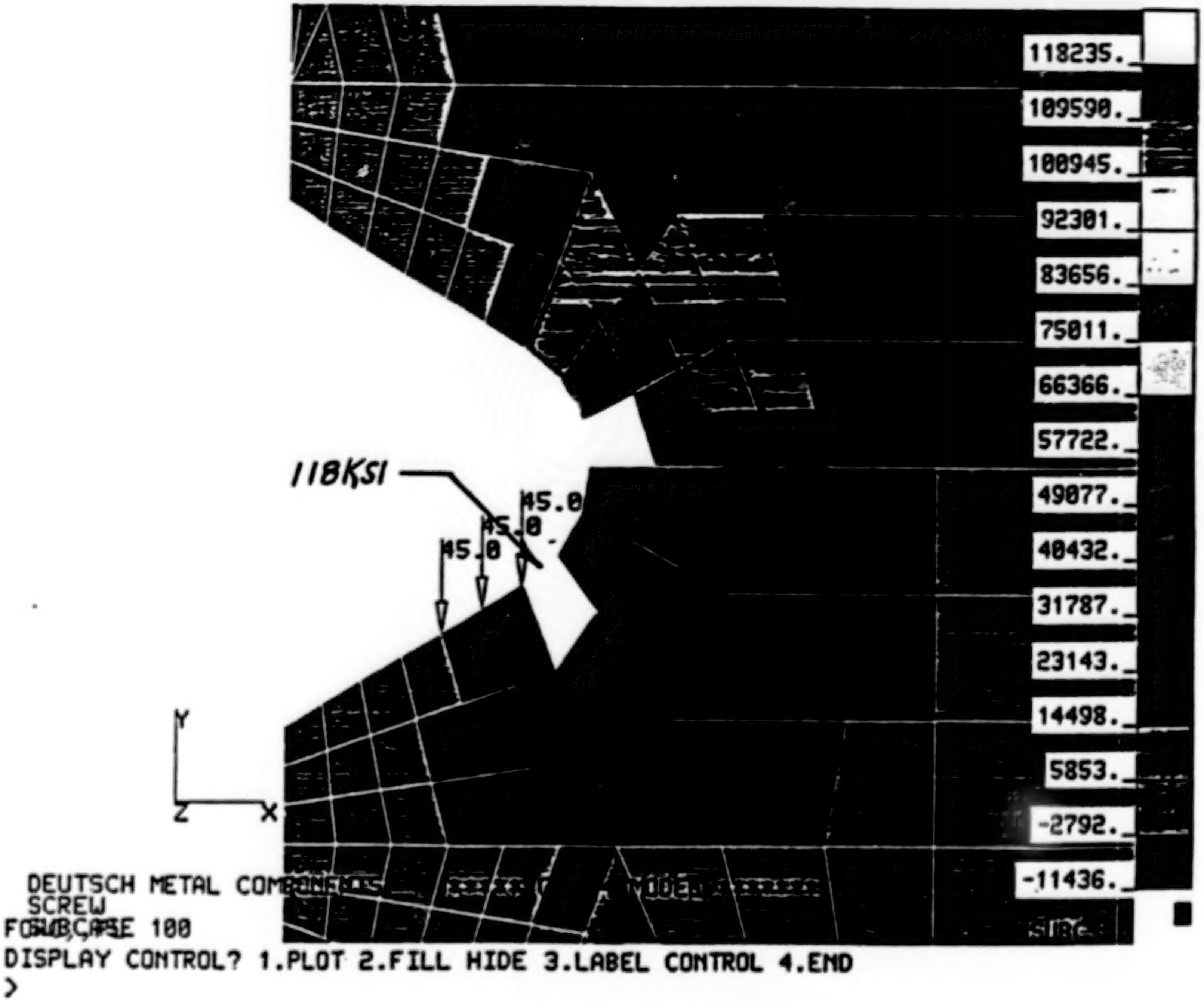
BEST COPY AVAILABLE



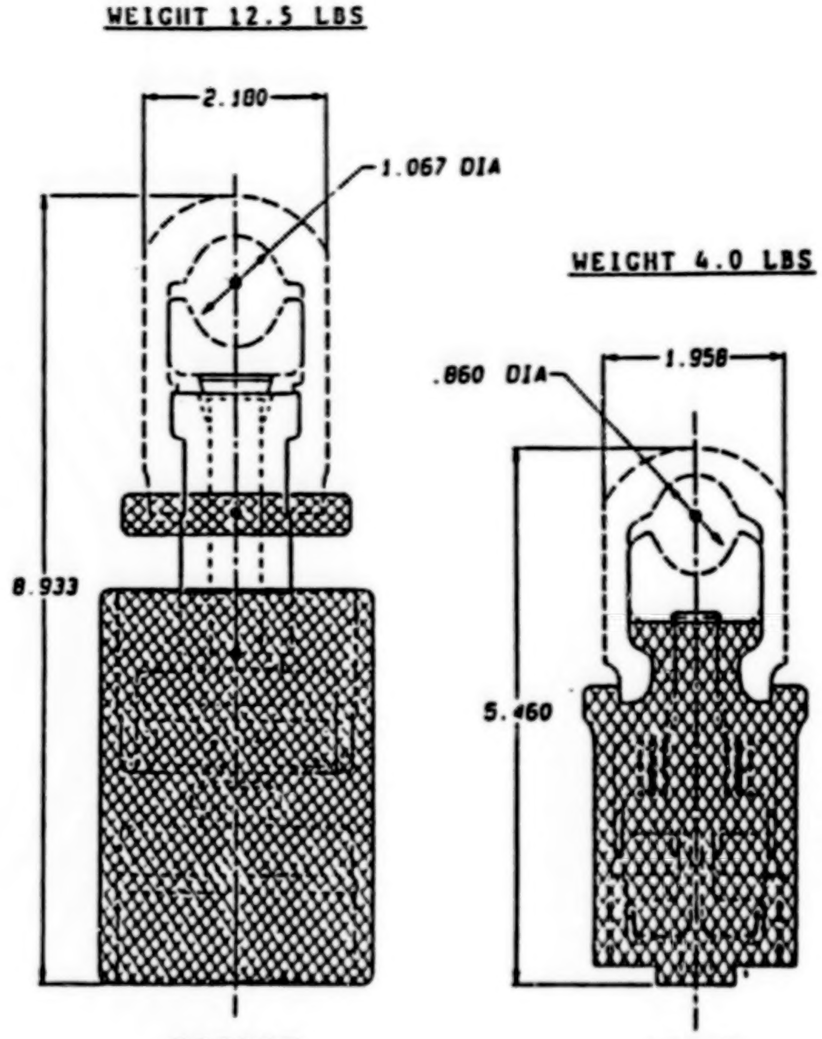
UNJ CYLINDER THREAD WITH
TAPERED THREAD LOADS APPLIED

DEUTSCH METAL COMPONENTS ****** QUAD4 MODEL *******
SCREW
SUBCASE 100
[illegible] ANY CHARACTER TO SELECT
WINDOW? 1.ZOOM 2.MOVE 3.CENTER 4.FIND 5.CORNERS 6.RESTORE 7.EXIT
>

233
13
13
13
Y
Z
X
118235.
109590.
100945.
92301.
83656.
75011.
66366.
57722.
49077.
40432.
31787.
23143.
14498.
5853.
-2792.
-11436.
SUBC


118KSI
45.0
45.0
45.0
118235.
109590.
100945.
92301.
83656.
75011.
66366.
57722.
49077.
40432.
31787.
23143.
14498.
5853.
-2792.
-11436.
Y
Z
X

```
DEUTSCH METAL COMPONENTS
SCREW
SUBCASE 100
DISPLAY CONTROL? 1.PLOT 2.FILL HIDE 3.LABEL CONTROL 4.END
>
```



BEFORE AFTER

INAL DESIGN

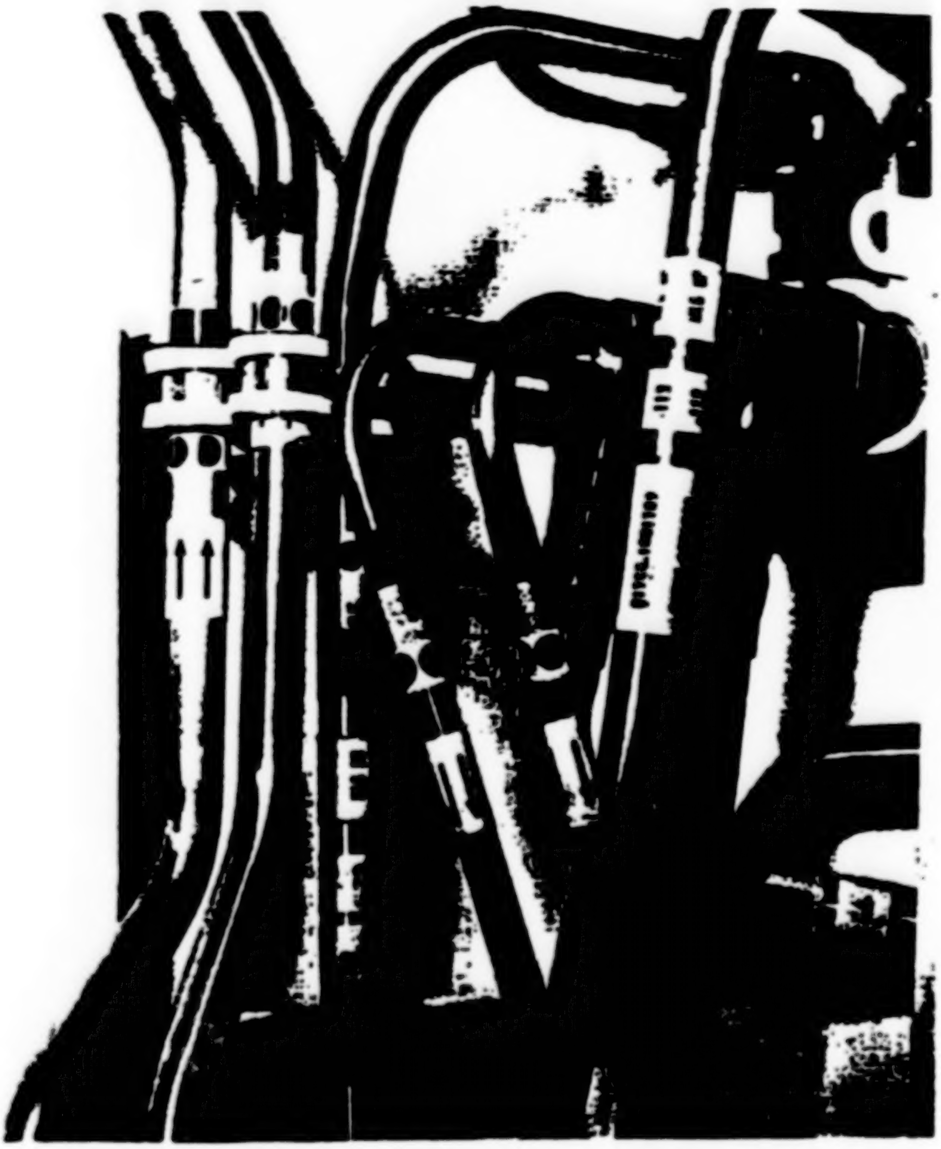
e above slide shows the "before and after" configuration after incorporating the aforementioned anges. The right hand slide shows the final design.

e 2 units shown have the same indentical swaging force and swage the same size fitting. The erall height had changed from approximately 10" to 5½". The cylinder diameter from 3.235 to 67 diameter. The head width from 2.180 to 1.958 and the weight from 12.5 pounds to 4.0 unds.

is new design with its fewer and smaller parts has increased, in numerous areas, the profitability d producibility of this product line.

BEFORE

AFTER



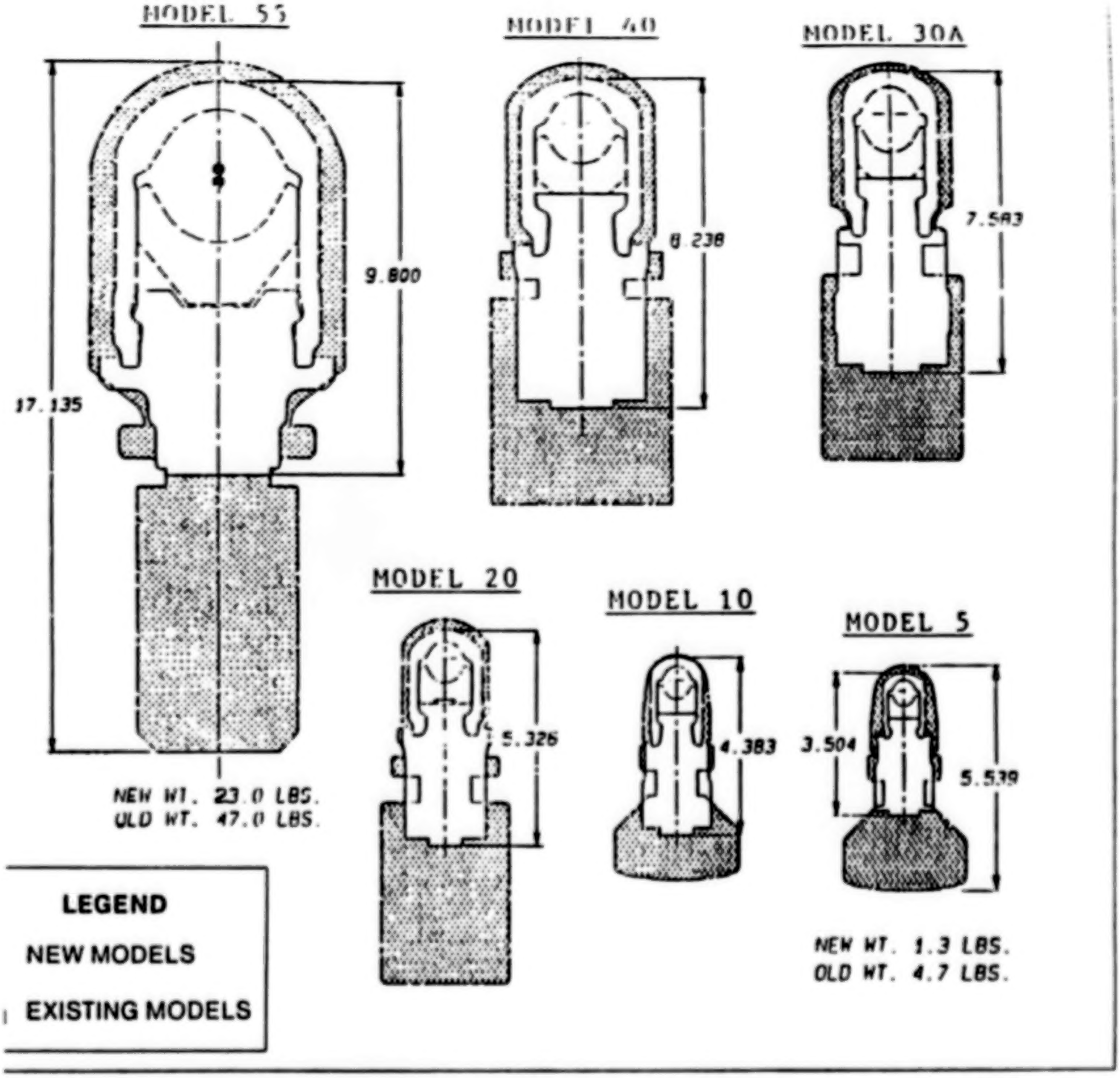
IMPORTANCE OF SIZE & WEIGHT

This smaller size design which we were able to generate was very important to the marketing department. The smaller configuration allowd the tooling to perform the swaging operations in very confined and cramped areas such as would be found in military aircraft and piping banks aboard ship. While the original objective was to increase operational cycle life it soon became very evident that the smaller size and weight was a definite plus in marketing this product.

ENGINEERING INNOVATIONS

Although, finite element analysis played a large part in accomplishing the design objectives there were some engineering innovations which also contributed to the success of this product. These tems include the slide-on type head feature, the material selection, the tapered thread and the change from 5500 psi to 10,000 psi hydraulic pressure. All other changes were directly attributed to inite element analysis.

BEST COPY
AVAILABLE


MODEL 55
9.800
17.135
NEW WT. 23.0 LBS.
OLD WT. 47.0 LBS.
MODEL 40
8.238
MODEL 30A
7.583
MODEL 20
5.326
MODEL 10
4.383
MODEL 5
3.504
5.539
NEW WT. 1.3 LBS.
OLD WT. 4.7 LBS.
LEGEND
NEW MODELS
EXISTING MODELS

COMPLETE PRODUCT LINE

he existing models are shown in the shaded area. The new models are shown in the white area ithin the shaded areas.

SUMMARY

e started with the design objective to increase the operational cycle life of the Swaging Tool. To ccomplish this increase in cycle life without increasing the size or weight of the tool would have een engineering achievement. However, we not only incrased the operational cycle life between 2 o 10 x but simultaneously we decreased the size and weight of the Swage Tool by about 50%. This ccomplishment now becomes an outstanding engineering achievement.

his achievement was only possible because of the computerized Patran, Nastran and Medusa rograms.

BEST COPY
AVAILABLE

OFTWARE & HARDWARE USED FOR PROGRAM

ie analysis shown in this paper was performed at the Deutsch Metal Components Division using
e **Patran & Cosmic/Nastran** programs. The geometry was generated on Medusa Program and
ansferred to Patran for analysis. The computer used was a **Prime 2655.** Hardcopies of the finite
ement model and analysis results were obtained by Patran through a Tektronix 4115 terminal
ooked to a Tektronix 4692 Ink-jet plotter.

EFERENCE:

oeckly, E.E., and Macke, H.I., "Effect of Taper on Screw-Thread Load Distribition." Transaction
74, A.S.M.E., 1952, pg. 103.

The Use of Cosmic Nastran in an Integrated Conceptual Design Environment

By Gil White, Intergraph Corporation

Conceptual engineering is increasing with the advent of the engineering workstation as a viable platform for numerical analysis, including finite element analysis. Traditionally, engineers have used finite element analysis after the detailed design stage had begun. There were exceptions in the area of load path models which were used early in the design process. New hardware platforms and software techniques now bring tools for finite element analysis into the mainstream conceptual design phase. A survey by a major British aerospace firm determined that the first five percent of design time dedicates an astounding eighty percent of the project cost. By using **COSMIC NASTRAN** early in the design phase, the total project cost can be reduced. The development of automated meshing routines that work within **COSMIC NASTRAN** pre processors also reduce the cost associated with finite element analysis and helps bring this tool into the conceptual design environment. Even though specialized finite element analysis should be reserved for professional engineers, there is a place for less experienced users in this area. The development of advanced meshing routines also allow the user to have confidence in the finite element mesh. Many systems also optimize the element shapes. Other features that are becoming popular with engineers and designers are adaptive refinement and geometry based analysis. Both of these are made possible by systems that have a common database for design, engineering and manufacturing. By using this same database, the finite element analysis does not have to redefine the model. This reduces the chance for errors and helps bring a product to market faster. These features are bringing the use of finite element analysis into mainstream mechanical conceptual design.

Intergraph Corporation has recently developed a suite of tools for mechanical computer aided engineering (MCAE). These tools break from the traditional design - analysis relationship in that both design and engineering data are retained in a single database. This concept, which is termed "geometry-based analysis," makes the analytical FEA model an extension of the design geometry. To the **COSMIC NASTRAN** user, this means all model attributes such as loads, boundary conditions, materials and properties are assigned to the CAD design geometry before finite element analysis begins.

These features are made possible by a system that differs structurally from Computer Design and Analysis systems of the past. The basis for the design - analysis relationships described above is a unique design tool named The Intergraph Engineering Modeling System (I/EMS). Built onto I/EMS are application specific tools for finite element analysis, mechanism analysis and other engineering-specific requirements. In the area of finite element analysis, I/FEM is built directly onto I/EMS. I/FEM is a complete system for model generation, analysis and post processing as well as a full support system for **COSMIC NASTRAN**. Some of the features of I/FEM include automatic mesh generation and geometry-based analysis methods as mentioned above. Mesh generation within the I/FEM environment includes traditional single and semi-automatic, meshing as well as fully automatic meshing. With automatic mesh generation, the user can mesh an entire surface model with a single command, without extensive setup work. The mesher recognizes boundaries of different material, load, or property and places nodes and elements at the proper locations. In addition the system performs smoothing operations that result in correctly shaped elements. At all times, the user has control over element shape criteria and is warned when rules are violated. I/FEM is the only finite element system system available with all of the following features:

- Object oriented programming concepts.
- A common mathematical description for all geometric entities, the Non-Uniform Rational B-Spline (NURB).
- The ability to generate surface, solid and wireframe entities within a single model.
- A Relational Data Base.

These four features provide the basis for design - analysis relationships within the Intergraph environment. Object oriented programming allows a graphic entity such as a surface or solid to know its material composition. An "object" can be defined as entities that contain within themselves both the information that defines how they behave (action) and information that defines their existence (state). Within an object-oriented software system, this means that an object is a package (in memory) of data and procedures that go together. In an object-oriented software environment, a subclass inherits all of the instance variables, methods, and message protocols of its superclass. To specialize from a class, one merely creates a subclass, adding additional instance variables, methods and messages only as needed to define what is different between the new subclass and its superclass. A subclass may also choose to override a method which it inherits, if the overridden method performs differently that the superclass method, based on some difference between the two classes. This powerful concept has far reaching potential. The finite element engineer can now receive an intelligent design geometry that includes many modeling attributes. When combined with automatic meshing capabilities, the result is reduced model generation time. In addition, these features add a new dimension to adaptive refinement and optimization.

Beyond the benefits in the traditional engineering areas, I/FEM offers its greatest potential in the conceptual design phase. The use of engineering tools in this phase of design is considered by many to be one of the ways American manufacturing will again become competitive in the world market. To bring tools like finite element analysis into mainstream mechanical design the designer must be be given user friendly software tools. In many cases designers can perform preliminary analysis that can be very beneficial. The detailed analysis must of course be left to the finite element specialist. Intergraph has designed a user interface into I/FEM that greatly simplifies model generation. Other features such as adaptive refinement and element shape optimization give the designer more confidence in finite element analysis.

Another major factor that will determine to what extent finite element analysis will be used by design groups is hardware availability. Traditionally engineers have run finite element analysis on large computers. Recently engineering workstations have proven to be a viable platform for numerical analysis. All major finite element systems now run on engineering workstations. By migrating to such platforms the user has more control over the total process and in almost all cases has faster turnaround. **COSMIC NASTRAN runs on Intergraph's workstations.**

The geometry based application capability also applies to other areas of MCAE including mechanism and computational fluid dynamics. These concepts result in integration between different areas of engineering as well as between design and engineering. For example mass properties from a finite element model can be used directly by the mechanism model. The result is a system that brings the engineer closer to the design and manufacturing process which in turn reduces the overall cost associated with product development.

In summary, changes in both software and hardware are rapidly bringing conceptual engineering tools like finite element analysis into mainstream mechanical design. Systems that integrate all phases of the manufacturing process provide the most cost benefits. The application of programming concepts like object oriented programming allow for the "encapsulation" of intelligent data within the design geometry. This combined with declining cost in per seat hardware bring new alternatives to the user. Such systems are being offered by Intergraph today.

A NASTRAN/TREETOPS Solution to a Flexible, Multi-Body Dynamics and Controls Problem on a UNIX Workstation

Javier E. Benavente and Norris R. Luce
Dynacs Engineering Co., Inc.
Clearwater, Fla.

SUMMARY

Demands for non-linear time history simulations of large, flexible multi-body dynamic systems has created a need for efficent interfaces between finite-element modeling programs and time-history simulations.

One such interface, TREEFLX, an interface between NASTRAN and TREETOPS, a non-linear dynamics and controls time history simulation for multi-body structures, is presented and demonstrated via example using the proposed Space Station Mobile Remote Manipulator System (MRMS).

The ability to run all three programs (NASTRAN, TREEFLX and TREETOPS), in addition to other programs used for controller design and model reduction (such as DMATLAB and TREESEL, both described in this paper), under a UNIX Workstation environment demonstrates the flexibility engineers now have in designing, developing and testing control systems for dynamically complex systems.

INTRODUCTION

Traditionally, the "Modern" control design process has begun with a linearized representation of a model. From this point several paths may be taken to derive gains that form the basis of a feedback control system.

Many tools exist today that facilitate this control design process. One such tool, DMATLAB, accepts the model via the (A,B,C,D) matrices defined by;

$$\dot{x}(t) = A\, x(t) + B\, u(t)$$

$$y(t) = C\, x(t) + D\, u(t)$$

where:	$x(t)$	is the state vector	$x \in R^{nx}$
	$u(t)$	is the input vector	$u \in R^{nu}$
	$y(t)$	is the output vector	$y \in R^{ny}$
	t	represents time	$t \in R$
	A	is the state matrix	$A \in R^{nx \cdot nx}$
	B	is the control matrix	$B \in R^{nx \cdot nu}$
	C	is the state output matrix	$C \in R^{ny \cdot nx}$
	D	is the control output matrix	$D \in R^{ny \cdot nu}$

The question arises, where do the (A,B,C,D) system matrices come from? TREETOPS, a non-linear time history simulation for multi-body systems with active control elements, answers this question via a linearization option which produces the (A,B,C,D) matrices as an output.

TREETOPS numerically derives the equations of motion of systems based on a user defined topology. For rigid systems, the process is simple. The mass and inertia properties of each rigid body in the system is specified, along with node point geometry. The relationship between the bodies is specified by defining hinges. Sensors and actuators are easily included, along with controllers and other simulation elements.

For flexible bodies the topology is defined in a similar manner; however, additional modal data is needed for TREETOPS to accurately simulate the flexible system response [ref. 1]. Until recently, this flex data had to be generated off line and in a form compatible with TREETOPS.

The development of TREEFLX has allowed the use of NASTRAN to generate flexible models of the individual bodies represented in the TREETOPS system. TREEFLX utilizes the NASTRAN data to generate all of the terms required by TREETOPS to simulate the time-history response of a flexible, multi-body dynamic system.

This paper demonstrates the general modeling and control design process and the role NASTRAN plays within it. The paper is organized as follows. First, some comments on system observability/controllability and reduced order controller design is presented, along with comments on a general control design procedure. Next, the topology of the system of interest is presented and a rigid model of the system is developed to facilitate controller design. The controller is derived based upon the rigid system. With this analysis complete, the bodies are modeled as flexible via NASTRAN. For computational considerations, component model reduction is performed on the flexible model. The reduced order model is used to evaluate the controller designed with the rigid system.

It should be emphasized that all of the analysis, modeling and design work for this paper was completed on UNIX Workstations, namely, a SUN 3/60 and Silicon Graphics Personal IRIS Workstation. The ability for an engineer to model a complex multi-body flexible system with a complete version of NASTRAN, design a controller for that system and simulate the non-linear closed-loop time history on a relatively inexpensive UNIX Workstation is a major advancement in computer aided engineering analysis and design.

CONTROL DESIGN CONCEPTS

It is generally acknowledged by control designers that the model and control design processes are inseperable. Indeed, Skelton [ref. 2] refers to this as the Modeling and Control Inseparability Principle. Simply put, the modeling and control designs are necessarily iterative.

Often, simple models of a physical systems are employed to facilitate the control design. As an example, consider a single beam modeled as a flexible body by NASTRAN. Suppose 20 flexible modes are retained for the TREETOPS representation of this beam, and that one rigid rotational degree of freedom is provided by a pinned hinge. Further, suppose that two sensors, one to measure the hinge angle and the other to measure the rate of change of the angle, along with a torque actuator, are co-located at this hinge. The linearized TREETOPS state matrix would be size 42 by 42. Controllability and observability (in the sense of a Linear Quadratic (LQ) control design) is certainly not guaranteed and probably not likely.

Now consider the body as rigid, with a stiff spring placed at the pinned hinge to approximate the body's flexibility. For this model the TREETOPS state matrix is size 2 by 2. In general, observability is guaranteed and controllability is more likely; a solution to the LQ design problem is, in general, easier to obtain with simpler models.

As Anderson and Liu mention [ref. 3], the above process is in reality a crude, yet sometimes successful, approach to controller reduction.

A logical question to be asked here is: How does the performance of a controller based on a simple model of a system change when applied to a more complex representation of the same physical phenomenon?

Figure 1 shows the control design process used in this paper. Below each process block is the name of the computer program(s) utilized in this paper to accomplish the process' objectives. Figure 2 shows the general relationship and interaction between these programs as implemented in a UNIX Workstation environment.

This paper demonstrates the design process of Figure 1 by example. A simplified lumped flexibility model of the MRMS is developed to form the basis of an LQ controller. Once settled upon, this controller is applied to a more complex system derived from NASTRAN models. Performance characteristics are compared between the two models.

MODEL TOPOLOGY

Figure 3 shows the general topology of the system of interest in this paper, a model of the MRMS. Represented is a 4-body system, the first and fourth bodies both being rigid, the second and third bodies both flexible.

Two sensors each are located at the second and third hinges. The first sensor measures the Euler angle between each hinge's inboard and outboard body, the second sensor measures the rate of change of the angle. A torque motor actuator is co-located with the sensors at both of the hinges. For simplicity, only one rotational degree of freedom is modeled at both the second and third hinge. All other hinges are locked. Physical properties of the individual bodies are summarized in Table 1.

The control design objective is to minimize perturbations from the initial conditions of the Euler angles, as measured by the sensors at the second and third hinges, in the presence of a disturbance. The disturbance is modeled within TREETOPS by a non-periodic pulse acting at the end of the third body.

CONTROLLER DEVELOPMENT

To facilitate control design, a lumped flexibility model is developed with the aid of TREESET and TREETOPS. This lumped flexibility model treats each body as being rigid; flexibility is modeled by lumping the body flexibility at the hinges with stiff springs.

The lumped flexibility model is entered into TREETOPS via TREESET, an interactive setup program. The linearization option is chosen and the simulation is set to run for one time step. Running the TREETOPS simulation results in a file containing the linear, time-invariant (A,B,C,D) matrices for the lumped flexibility model. This process is equivalent to Step 1 of Figure 1. Table 2 list the numerical values of the (A,B,C,D) matrices as output from TREETOPS' linearization option.

These matrices are entered into DMATLAB. DMATLAB provides many controls analysis and design tools for both the "classical" and "modern" controls designer. DMATLAB is used to design a controller based on the lumped flexibility model. This is equivalent to Step 2 of Figure 1.

Feedback control gains are obtained via an LQ control design based on full-state feedback. The state vector is defined by the two Euler angles and their rates. If we represent the angles by θ_2 and θ_3, their rates by $\dot{\theta}_2$ and $\dot{\theta}_3$ and the controller output (actuator commands) as u_1 and u_2 then the control law gain is a matrix G such that ;

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} G \end{bmatrix} \begin{bmatrix} \dot{\theta}_2 \\ \dot{\theta}_3 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

The numerical values for G will be found in Table 2.

TREESET is used to define a continuous matrix controller for the TREETOPS simulation. Interconnects are established between the sensors and the actuators, scaled by the gains determined in DMATLAB. This forms the basis of a continuous feed-back control system for the non-linear, time-history TREETOPS simulation. This simulation is the equivalent of Step 3 in Figure 1.

For complicated systems, an iteration for the controller gains will probably be required; indeed, the final controller gains for this paper were selected only after several such iterations.

NASTRAN MODEL

Since no official configuration for the MRMS has been established, the NASTRAN models are based on Space Shuttle Remote Manipulator System (SRMS) data [ref. 4].

The NASTRAN flex data is generated using CBAR elements to represent the mass and stiffness properties of SRMS body elements. Fixed-free boundary conditions were chosen for each body. Standard Solution 3 (Normal Modes Analysis), with an DMAP alter added for the additional output required by TREEFLX, is utilized. A separate OUTPUT5 file is generated for each flexible body in the topology. The development of the NASTRAN model is equivalent to Step 4 of Figure 4.

TREEFLX, based on the NASTRAN data for each body, calculates all of the required and optional modal data for the TREETOPS simulation. Table 3 summarizes the modal terms presently generated by TREEFLX and used by TREETOPS.

To generate this data, TREEFLX requires the NASTRAN Nodal Mass, Eigenvectors, Modal Mass, Modal Stiffness and, if available, Modal Damping matrices. In addition, a matrix consisting of NASTRAN Grid Point Location vectors, expressed in global coordinates, is required. The process of converting NASTRAN output data to TREETOPS input data with TREEFLX is represented by Step 5 of Figure 1.

A major assumption in TREEFLX is that the TREETOPS and NASTRAN models use the same coordinate system for each individual body. Based on this assumption, it is not necessary to designate the TREETOPS node location with coordinates during the TREESET setup procedure, but rather, the user designates a corresponding NASTRAN grid point ID for each TREETOPS node. TREEFLX uses this node/grid point correspondence to develop the TREETOPS nodal geometry. Not all NASTRAN grid points have to be included in the TREETOPS model. TREETOPS nodes are required only as hinge attach points, sensor and actuator locations and for mass centers.

An important distinction must be made at this point. Notice in Table 3 that several TREETOPS terms are calculated with summations over the total number of nodal bodies in the model. Even though all nodes may not be included in the final TREETOPS data file, the TREEFLX nodal summations are made over the entire set of NASTRAN grid points supplied in the NASTRAN OUTPUT5 file, not just over the sub-set of retained TREETOPS nodes.

COMPONENT MODE MODEL REDUCTION

An optional step may be inserted between Steps 5 and 6 of Figure 1. Suppose the complex model developed by NASTRAN includes 100 modes for each body, yet it is determined that a model with 47 modes for each body is sufficient for an accurate time-history simulation (this paper does not propose any method for this determination). This implies that a model reduction procedure might be inserted at this point of the design process.

TREEFLX provides for model reduction with a simple mode selection technique. If model reduction is indicated, TREEFLX searches for a file that lists the modes that should be retained for each individual body.

A natural question arises: Which modes should be retained in a reduced order model? This paper does not present a theoretical discussion of component mode model reduction procedures; however, TREESEL, a TREETOPS companion program, can assist in the answering of the above question.

TREESEL uses several methods to rank the relative importance of the system modes. One method, used in this paper to reduce the MRMS NASTRAN model, is the Modified Component Cost method.

The Component Cost method is based on the assumption that each state contributes to a cost function, $\mathcal{V}$, defined by the model designer. By decomposing the cost function into its components, the relative contribution of each system state to the cost function can be ranked.

In TREETOPS, each degree of freedom is a state. A beam with N flexible modes will have at least 2*N states, 2 states for each mode. TREESEL ranks these modal degrees of freedom in a concise form. Once ranked, the number of modes to be retained depends on the open loop performance matching the analyst would like to obtain. An iterative process of selecting modes is usually required to obtain a suitable reduced order model. Table 4 lists the NASTRAN modes as ranked by TREESEL. This ranking represents only a single iteration with TREESEL using simply selected weights.

To demonstrate TREESEL model reduction techniques, the five highest ranked modes for each body (10 system modes) were retained for the "complex" NASTRAN/TREETOPS model.

Step 5 of Figure 1 is accomplished by merging the NASTRAN/TREETOPS model with the continuous matrix controller gains derived earlier. TREETOPS is used to simulate the closed-loop time-history response of the system.

RESULTS

Figures 4 and 5 show the results of Steps 1, 2 and 3 of Figure 1. Plotted is the time-history of the uncontrolled vs controlled hinge Euler angles and rates for the lumped flexibility model. The results shown were considered adequate to accept the controller design.

Figures 6 and 7 show the results of component mode model reduction using the five highest body modes ranked by TREESEL. Shown are the uncontrolled hinge Euler angles and rates for the full-order (20 modes) and reduced-order (5 modes) NASTRAN model. The TREESEL ranking was obtained with just one run of the program and simply selected weight were used. The results seem to indicate that reduced order models of higher order systems can approximate the higher order system's uncontrolled response.

Figures 8 and 9 show the results of Step 6 of Figure 1. Plotted is the time-history of the uncontrolled vs controlled hinge angles and hinge angle rates for the NASTRAN reduced-order model. Figures 10 and 11 compare the uncontrolled responses of the Lumped Flexibility and NASTRAN reduced order models. Figures 12 and 13 compare the controlled responses of the Lumped Flexibility and NASTRAN reduced order models. The results indicate that, for some systems, controllers designed on the basis of simplified models of complicated systems do perform adequately on higher fidelity models of the same system.

Figure 14 compares the actuator commands (controller outputs) of the Lumped Flexibility and NASTRAN reduced order models.

CONCLUSIONS

This paper demonstrates the use of NASTRAN and a UNIX Workstation environment in the system modeling/control design process. An automated design environment on a UNIX Workstation applicable to modeling and control theory is presented.

TREEFLX is used to interface flexible body data from NASTRAN with the flexible multi-body non-linear analysis program TREETOPS. Powerful modeling and control design concepts are demonstrated via a non-trivial example. Results support the feasibility of using all of the programs in conjuction with one another to provide viable analysis and designs.

The ease in which the model or the controller can be changed further enhances the analysis turn-around-time and the design process itself, clearly demonstrating the advantages of working within a dedicated UNIX Workstation environment.

ACKNOWLEDGEMENTS

The authors wish to acknowledge and thank Jayant Ramakrishnan for his advice, suggestions and remarks. Thanks go to Mike VanderVoort for all the help he gave, Anren Hu, for advice on lumped flexibility models and Julie Hendricks, for all the help she gave, without which this paper would never of taken so long to finish.

Thanks go to John Sharkey for his suggestions on the improvement of the TREEFLX program. This paper was supported in part with funds from NASA MSFC contract number NAS8-37287.

REFERENCES

[1] TREETOPS Users Manual, Dynacs Engineering Company, Inc., 1988

[2] Skelton, Robert E., "Dynamic Systems Control: Linear Systems Analysis and Synthesis", John Wiley & Sons, 1988

[3] Anderson, Brian and Yi Liu, "Controller Reduction: Concepts and Approaches", Department of Systems Engineering, Australian National University

[4] Spar Aerospace Limited Document SPAR-R.775 Issue G, 1981, released by National Research Council of Canada for Unrestricted Use

TABLE 1. - MASS AND GEOMETRIC PROPERTIES OF MRMS MODEL

BODY	MASS	Ixx	Iyy	Izz	LENGTH
	(kg)		($kg\text{-}m^2$)		(m)
1	63.3	0	41.04	41.04	1.06
2	139.2	0	1877.9	1877.9	6.38
3	100.0	0	1429.9	1429.9	7.06
4	50.0	0	15.0	15.0	1.00

TABLE 2. - (A,B,C,D) MATRICES AND CONTROLLER GAIN MATRIX G

$$A = \begin{bmatrix} -.008816 & .016190 & -9.41087 & 11.2336 \\ .016190 & -.039067 & 17.28277 & -27.1063 \\ 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \end{bmatrix}$$

$$B = \begin{bmatrix} .000176 & -.000324 \\ -.000324 & .000781 \\ 0.0 & 0.0 \\ 0.0 & 0.0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \qquad D = \begin{bmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \\ 0.0 & 0.0 \\ 0.0 & 0.0 \end{bmatrix}$$

$$G = \begin{bmatrix} -5281.28 & 2243.41 & 1199.95 & -386.81 \\ 3478.13 & -8005.26 & -4361.25 & -1209.26 \end{bmatrix}$$

TABLE 3. - TREETOPS MODAL TERMS CALCULATED BY TREEFLX

$$\underline{h}_i = b^T \{ \sum_{o=1}^{NNB} [m_o \tilde{r}_o \{\phi_{oi}\} - \tilde{r}_o m_o \tilde{\ell}_o \{\phi'_{oi}\} + m_o \tilde{\ell}_o \{\phi'_{oi}\} + J^{bo} \{\phi'_{oi}\}]\}$$

$$\underline{a}_i = 1/_m \ \{ \sum_{o=1}^{NNB} m_o (\underline{\phi}_{oi} - \underline{\ell}_o \times \underline{\phi}_{oi})\}$$

$$\underline{\underline{M}}_i^b = b^T \{ \sum_{O=1}^{NNB} [-m_o \tilde{r}_o \tilde{\phi}_{oi} - m_o \tilde{\ell}_o \tilde{\phi}_{oi}]\} \ b$$

$$\underline{\underline{P}}_{ki}^b = b^T \{ \sum_{o=1}^{NNB} [-m_o \tilde{\phi}_{ok} \tilde{\phi}_{oi}]\} \ b$$

$$\underline{i}_{ki}^b = b^T \{ \sum_{o=1}^{NNB} [m_o \tilde{\phi}_{ok} \{\phi_{oi}\} - \tilde{\phi}_{ok} m_o \tilde{\ell}_o \{\phi'_{oi}\}$$

WHERE:

b represents the body reference frame

i,k represent the ith,kth modes

$\sum_{o=1}^{NNB}$ is the sum over the number of nodal bodies

m_o is the mass of the o^{th} nodal body

m is the body mass ; $m = \sum_{o=1}^{NNB} m_o$

CONTINUED

TABLE 3. - CONTINUED

J^{bo}	is the inertia matrix of the oth nodal body
$\underline{\phi}_{ok}$	is the kth mode shape at the oth nodal body
ϕ'_{ok}	is the kth mode slope at the oth nodal body
$\underline{\ell}_o$	is the vector location of the oth nodal body mass center
$\underline{r}_o$	is the vector location of the oth nodal body
$\{\cdot\}$	represents a 3x1 column matrix
$\tilde{\cdot}$	represents a skew symmetric matrix, that is, suppose $\underline{r}$ is given by; $\underline{r} = r_1 \hat{i} + r_2\hat{j} + r_3\hat{k}$ then $\tilde{r}$ is; $\tilde{r} = \begin{bmatrix} o & -r_3 & r_2 \\ r_3 & o & -r_1 \\ -r_2 & r_1 & o \end{bmatrix}$

TABLE 4. - TREESEL RANKING OF SYSTEM MODES (BY BODY)

RANK	BODY #2 MODES	BODY #3 MODES
1	14	16
2	9	4
3	11	2
4	4	9
5	6	11
6	2	6
7	13	18
8	19	14
9	8	5
10	5	3
11	10	8
12	3	1
13	18	10
14	7	17
15	12	7
16	16	13
17	20	20
18	15	15
19	1	19
20	17	12

This table presents data obtained with only one TREESEL iteration.
Simply selected weights were used.
The five highest ranked modes were retained
for the TREETOPS flexible model.

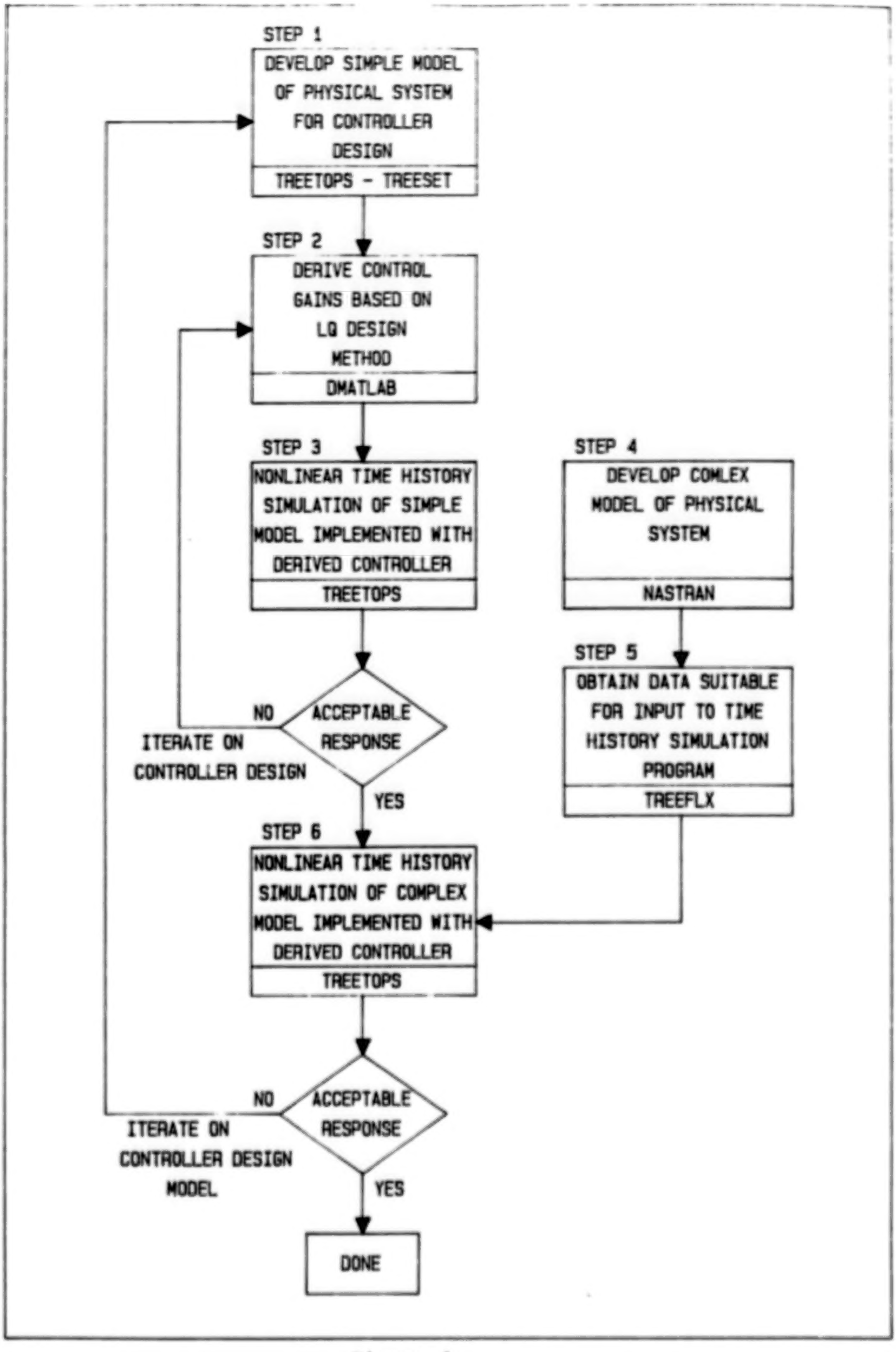

STEP 1
DEVELOP SIMPLE MODEL OF PHYSICAL SYSTEM FOR CONTROLLER DESIGN
TREETOPS - TREESET
STEP 2
DERIVE CONTROL GAINS BASED ON LQ DESIGN METHOD
DMATLAB
STEP 3
NONLINEAR TIME HISTORY SIMULATION OF SIMPLE MODEL IMPLEMENTED WITH DERIVED CONTROLLER
TREETOPS
STEP 4
DEVELOP COMLEX MODEL OF PHYSICAL SYSTEM
NASTRAN
STEP 5
OBTAIN DATA SUITABLE FOR INPUT TO TIME HISTORY SIMULATION PROGRAM
TREEFLX
NO
ACCEPTABLE RESPONSE
ITERATE ON CONTROLLER DESIGN
YES
STEP 6
NONLINEAR TIME HISTORY SIMULATION OF COMPLEX MODEL IMPLEMENTED WITH DERIVED CONTROLLER
TREETOPS
NO
ACCEPTABLE RESPONSE
ITERATE ON CONTROLLER DESIGN MODEL
YES
DONE

Figure 1.

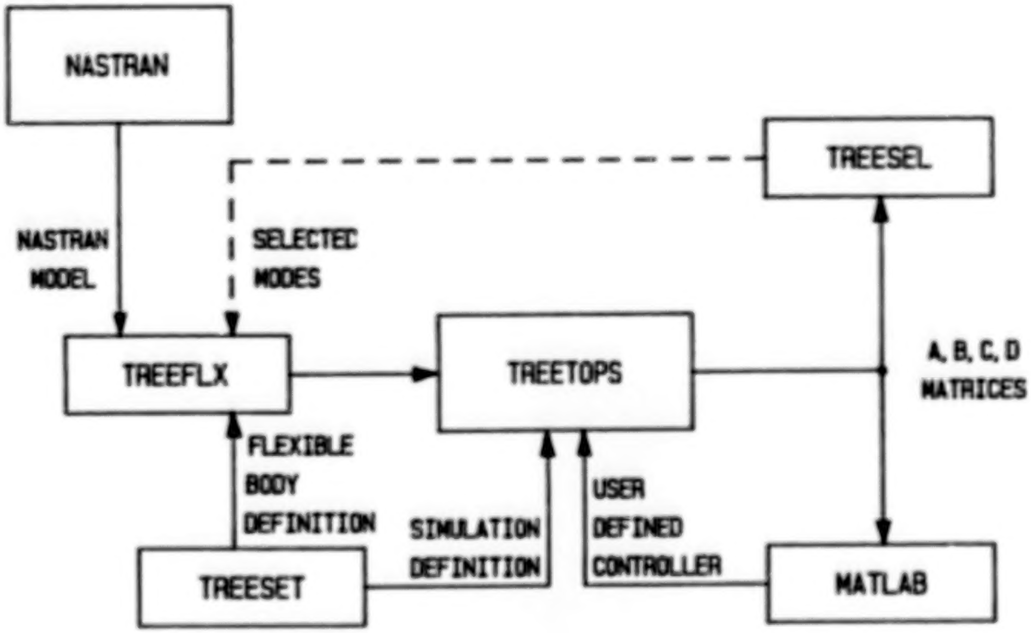

NASTRAN
NASTRAN MODEL
SELECTED MODES
TREESEL
TREEFLX
TREETOPS
A, B, C, D MATRICES
FLEXIBLE BODY DEFINITION
SIMULATION DEFINITION
USER DEFINED CONTROLLER
TREESET
MATLAB

Figure 2.

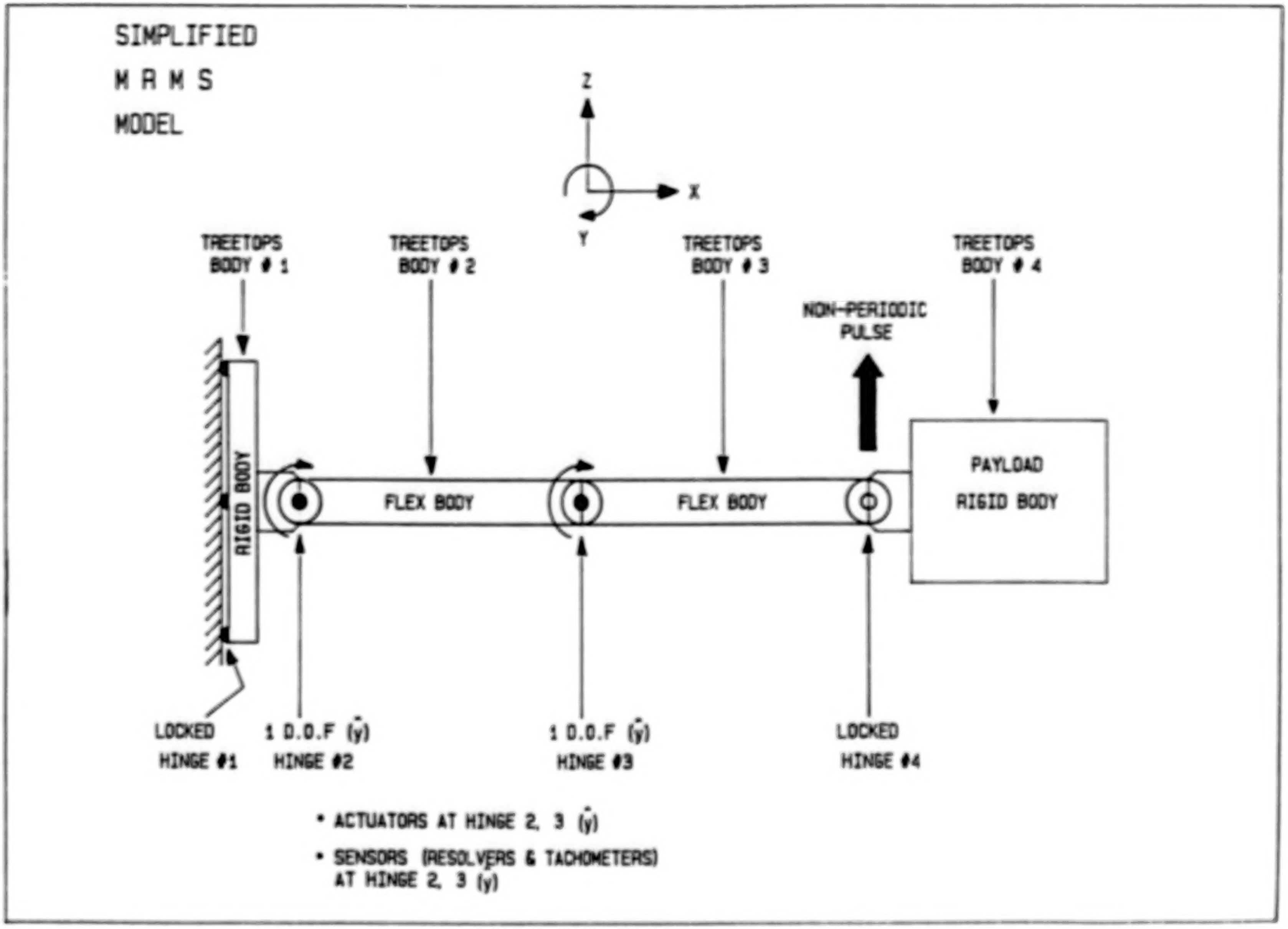

SIMPLIFIED
MRMS
MODEL
Z
X
Y
TREETOPS BODY # 1
TREETOPS BODY # 2
TREETOPS BODY # 3
TREETOPS BODY # 4
NON-PERIODIC PULSE
RIGID BODY
FLEX BODY
FLEX BODY
PAYLOAD RIGID BODY
LOCKED HINGE #1
1 D.O.F (ỹ) HINGE #2
1 D.O.F (ỹ) HINGE #3
LOCKED HINGE #4
• ACTUATORS AT HINGE 2, 3 (ỹ)
• SENSORS (RESOLVERS & TACHOMETERS) AT HINGE 2, 3 (y)

Figure 3.

Lumped Flexibility Model
Uncontrolled vs Controlled Response

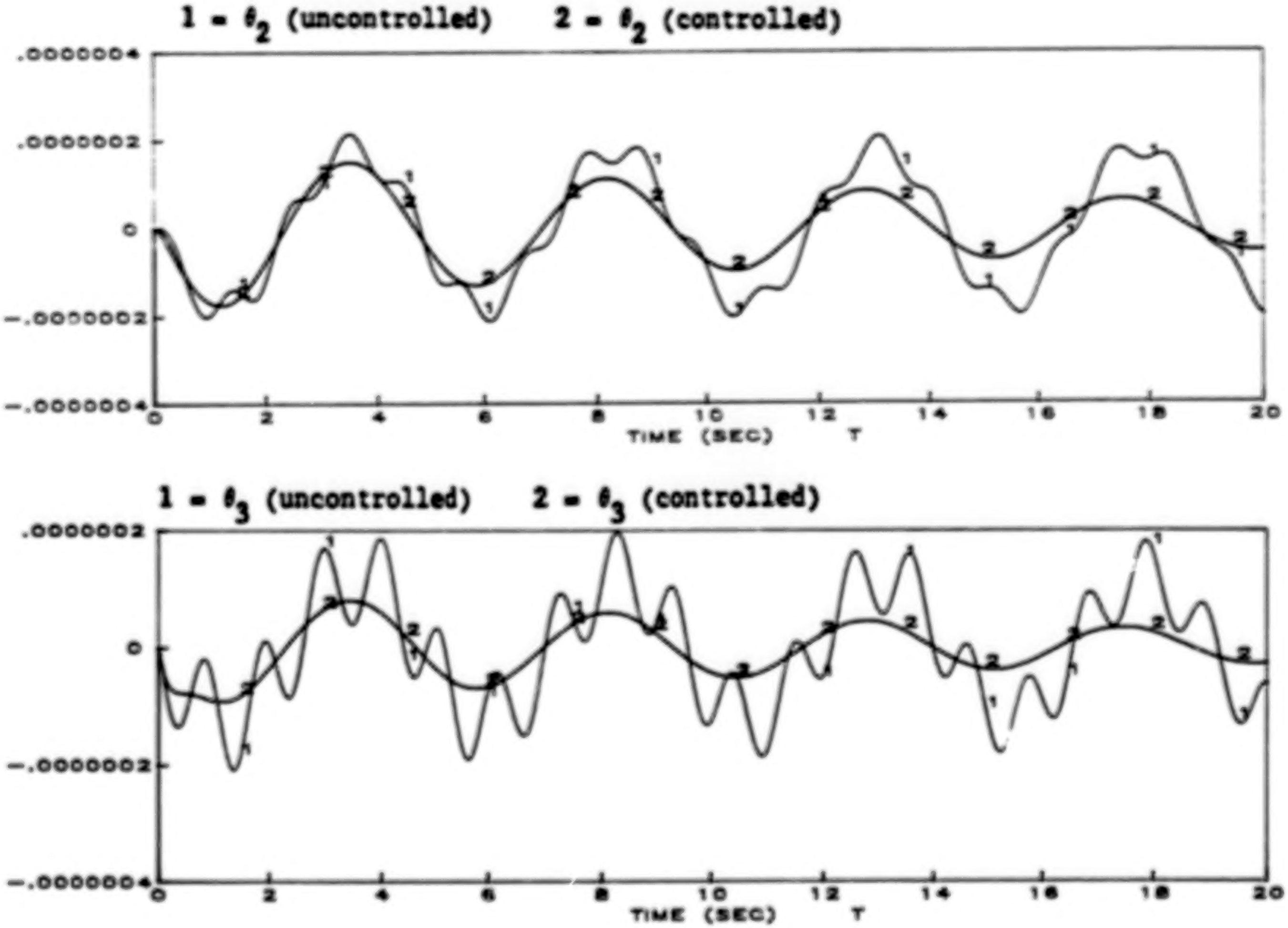


Figure 4.

BLANK PAGE

Lumped Flexibility Model
Uncontrolled vs Controlled Response

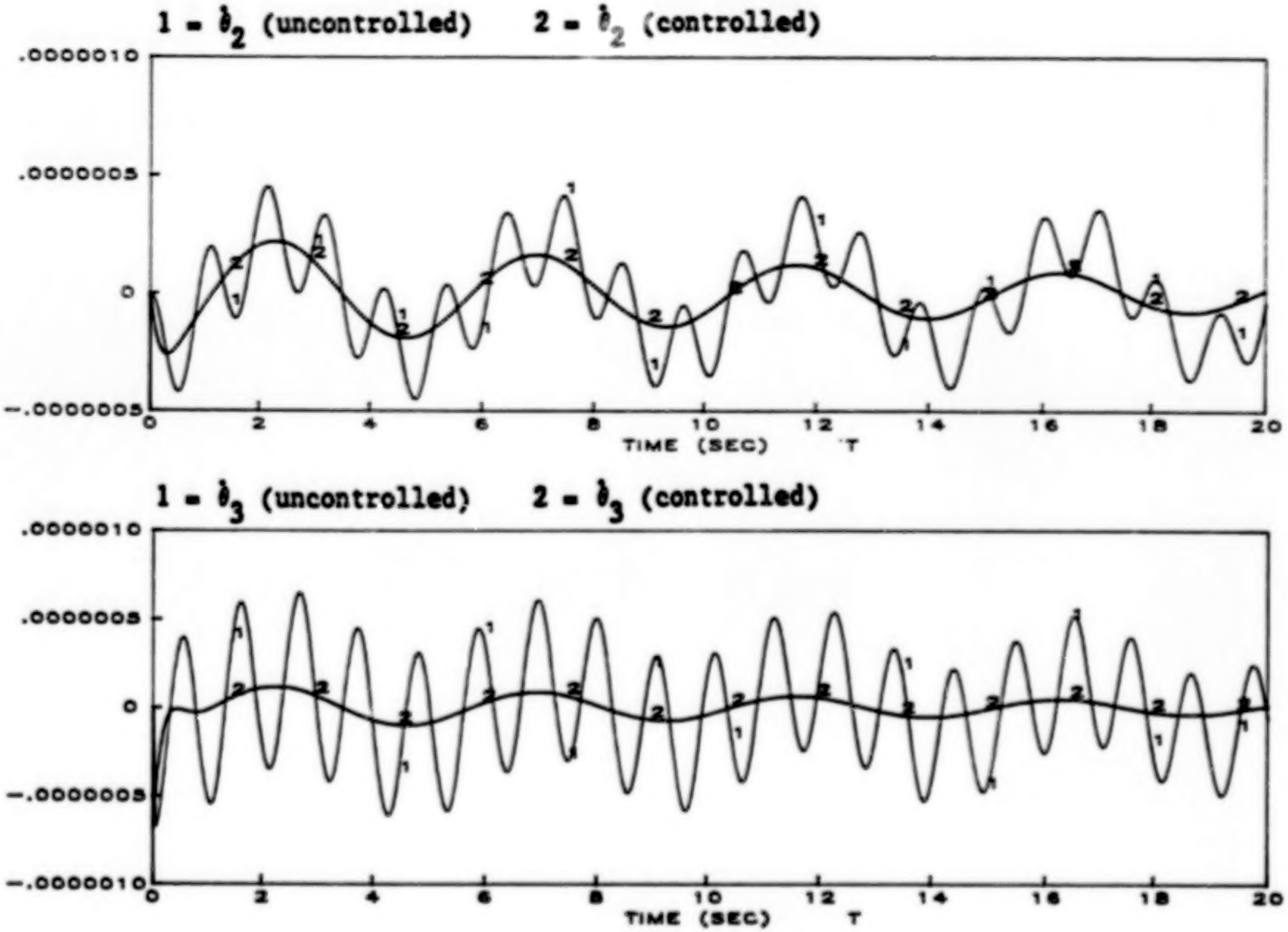


Figure 5.

Uncontrolled Response
Full Order (20 Modes) vs Reduced Order (5 Modes) Model

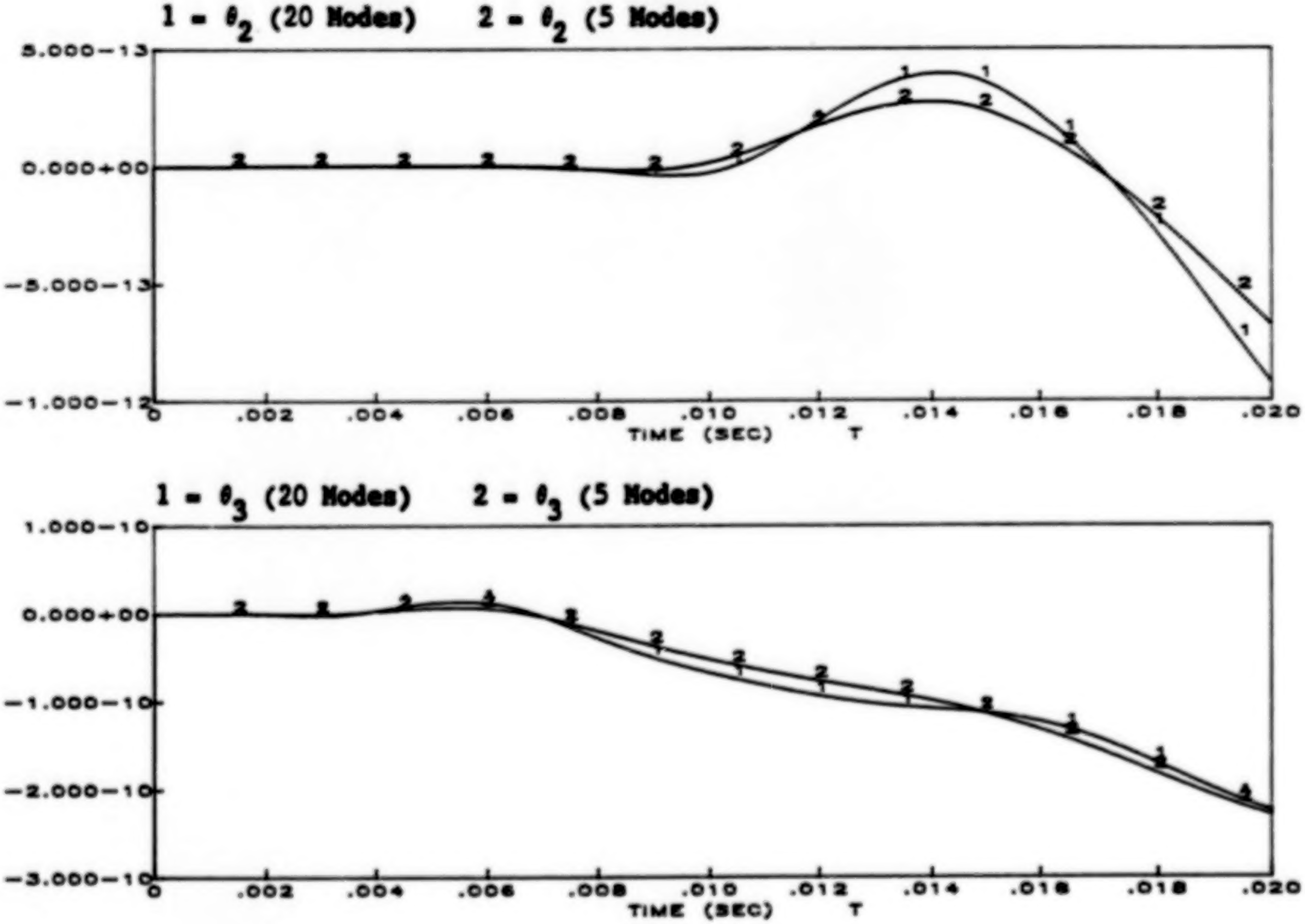


Figure 6.

Uncontrolled Response
Full Order (20 Modes) vs Reduced Order (5 Modes) Model

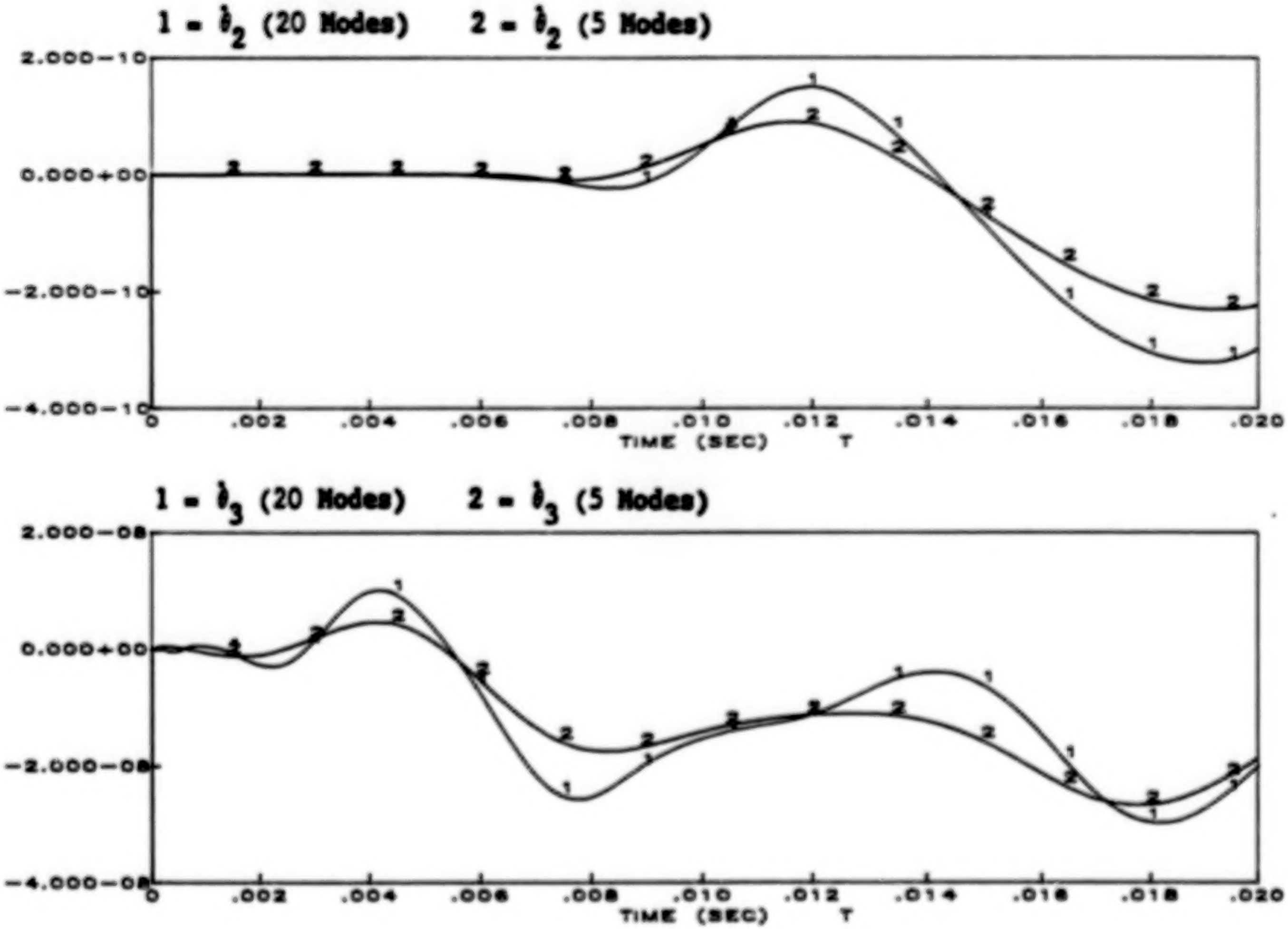
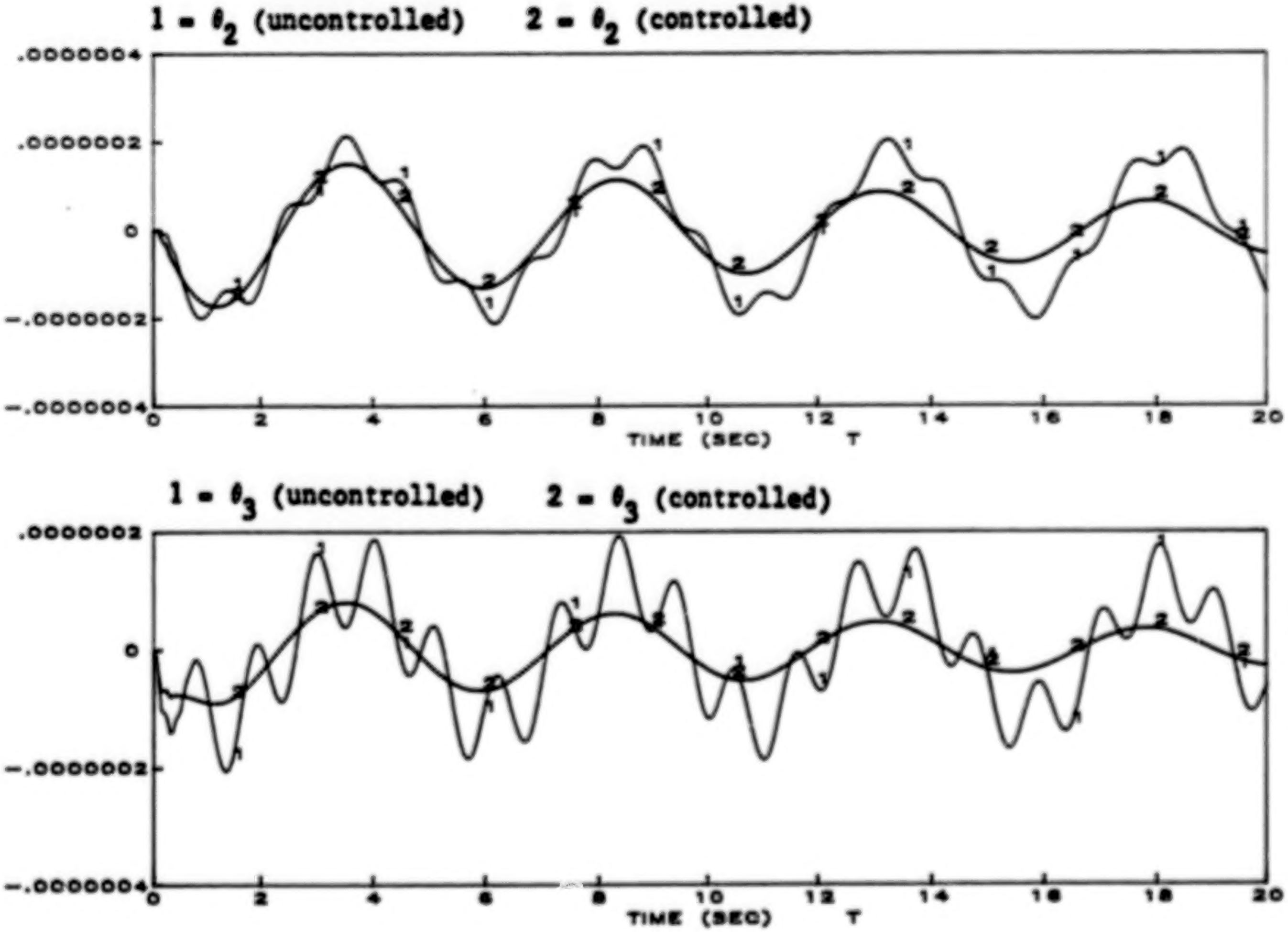


Figure 7.

NASTRAN Flexible Model
Uncontrolled vs Controlled Response


$1 = \theta_2$ (uncontrolled) $2 = \theta_2$ (controlled)

$1 = \theta_3$ (uncontrolled) $2 = \theta_3$ (controlled)

Figure 8.

NASTRAN Flexible Model
Uncontrolled vs Controlled Response

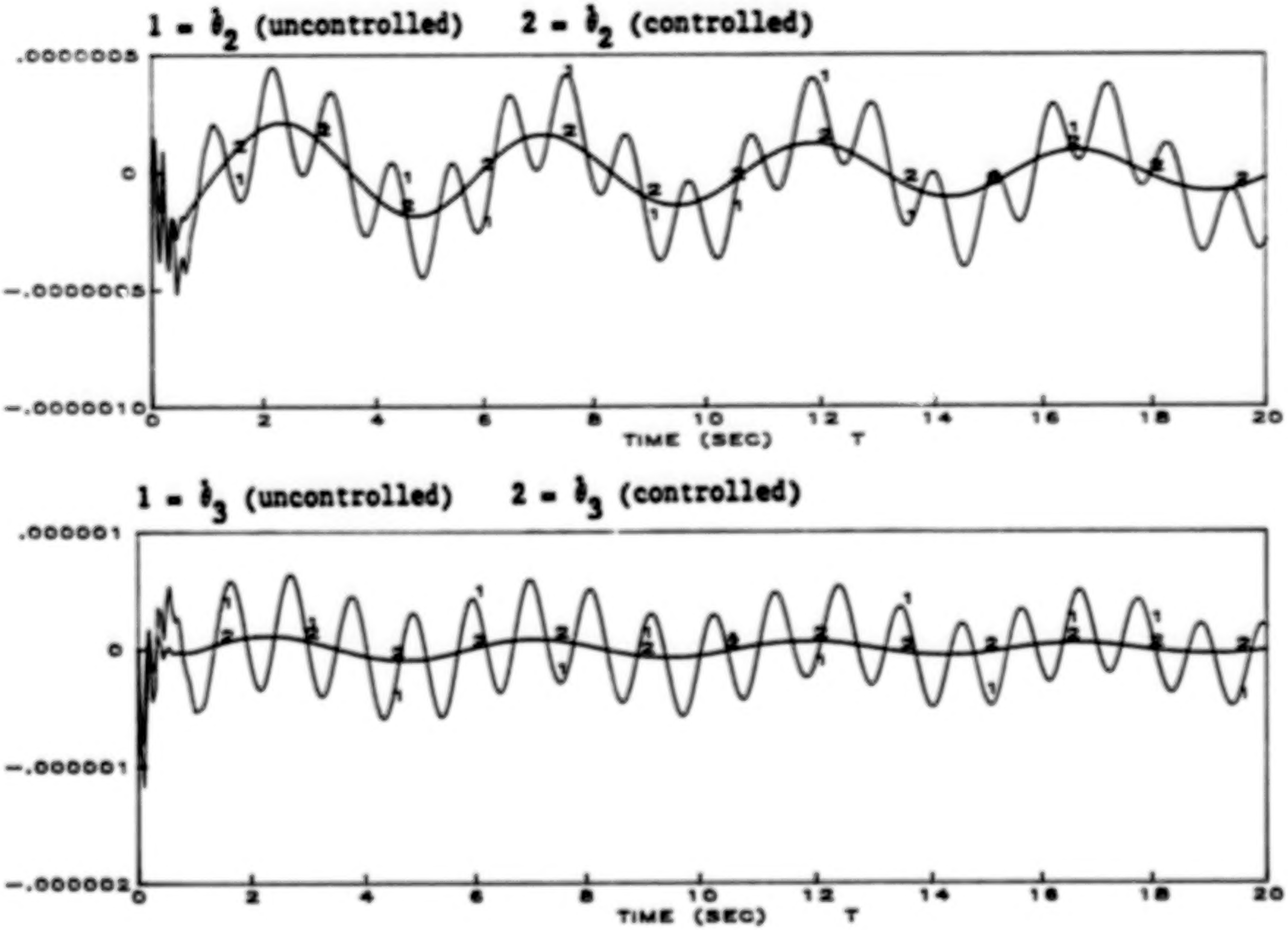


Figure 9.

Uncontrolled Response
Lumped Flexibility Model vs NASTRAN Model (reduced order model)

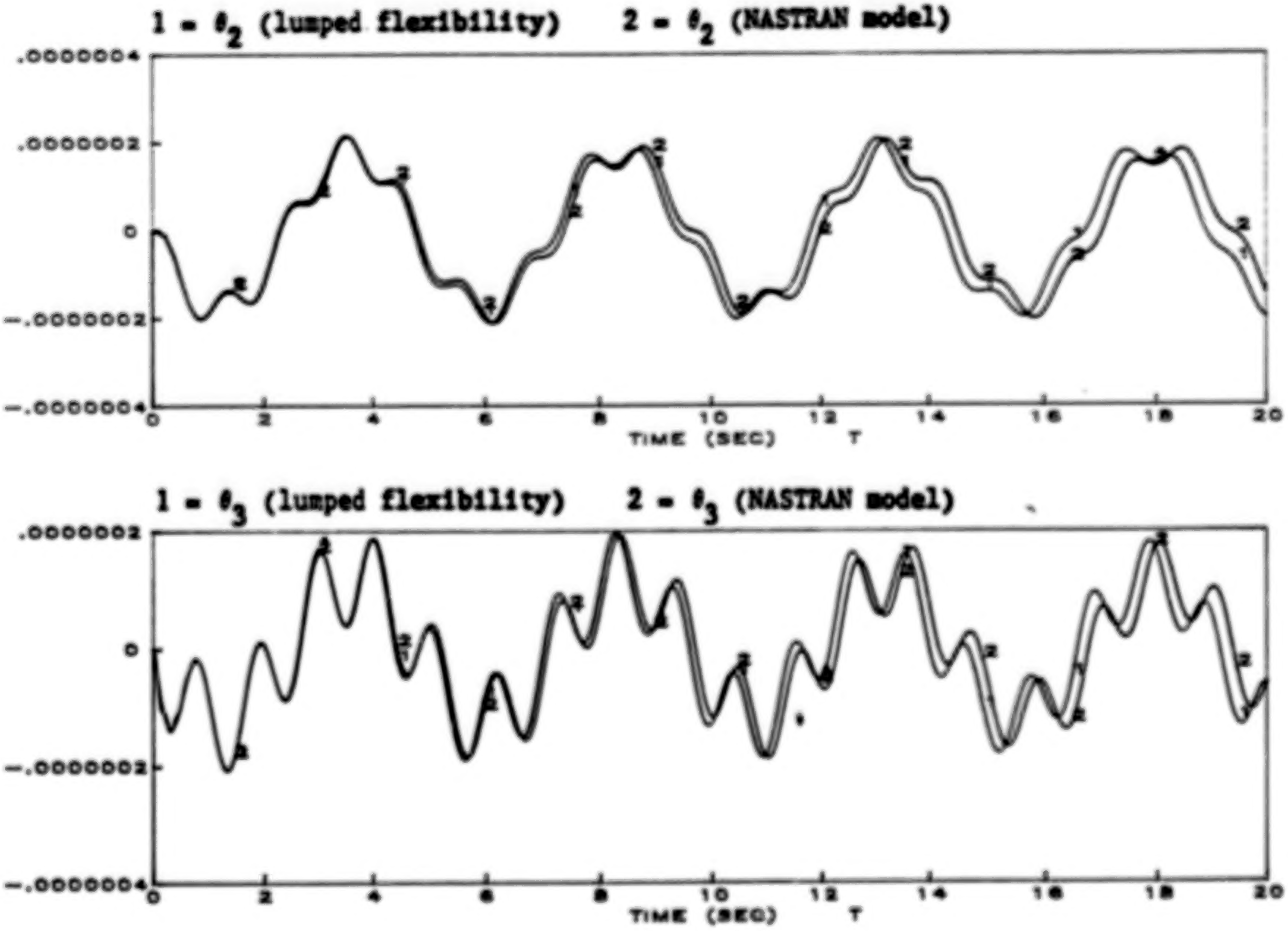


Figure 10.

Uncontrolled Response
Lumped Flexibility Model vs NASTRAN Model (reduced order model)

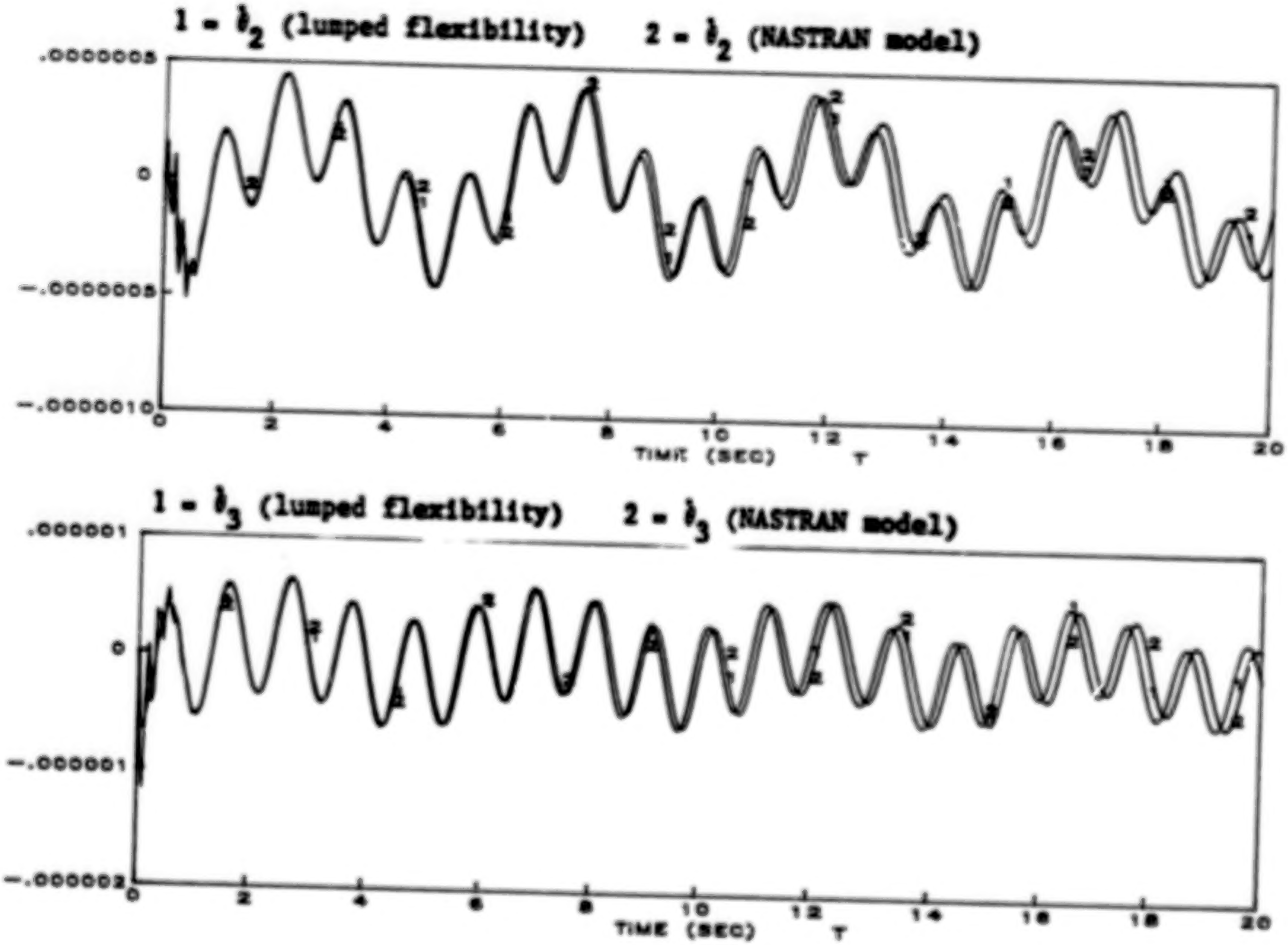


Figure 11.

Controlled Response
Lumped Flexibility Model vs NASTRAN Model (reduced order model)

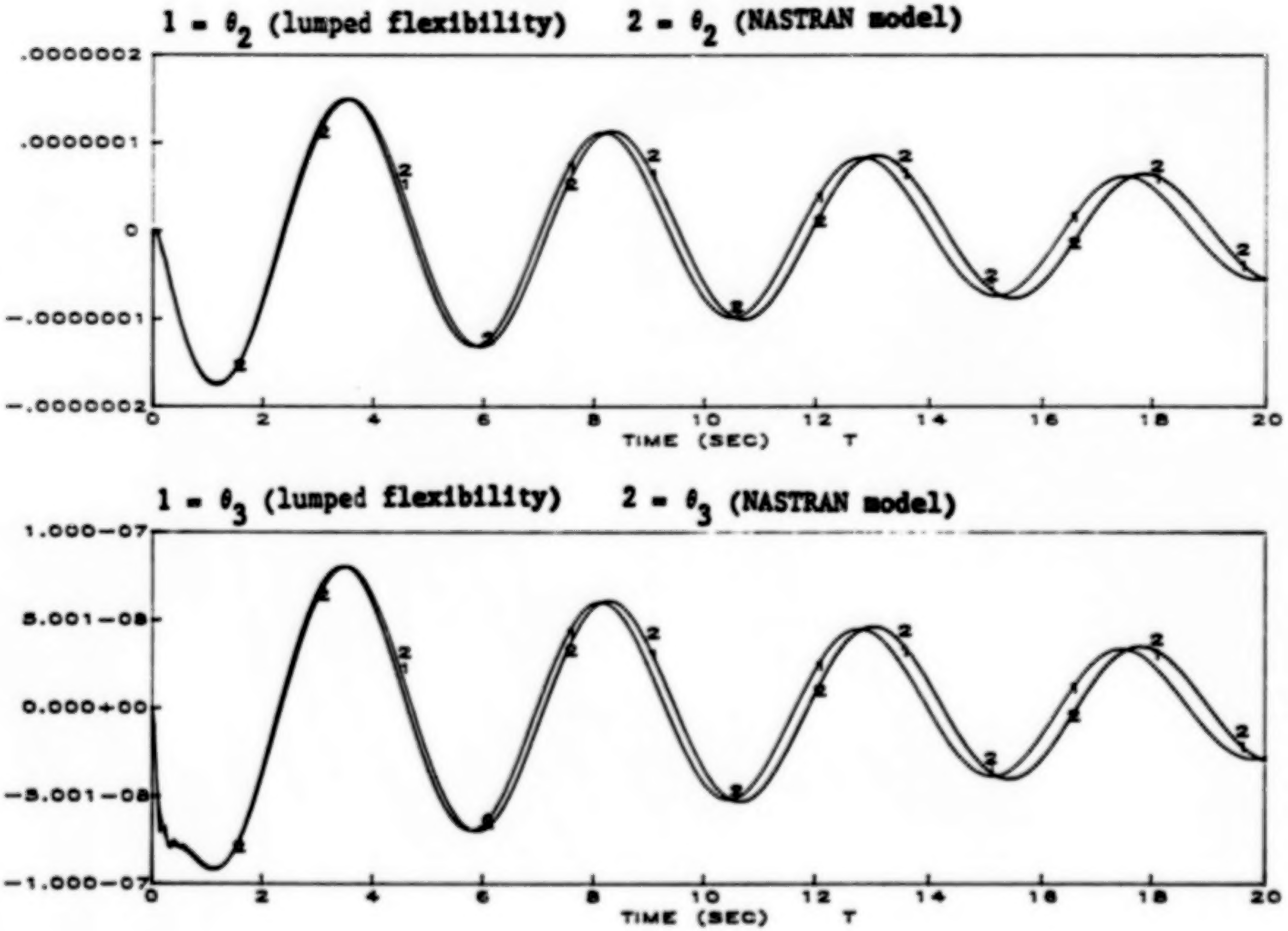
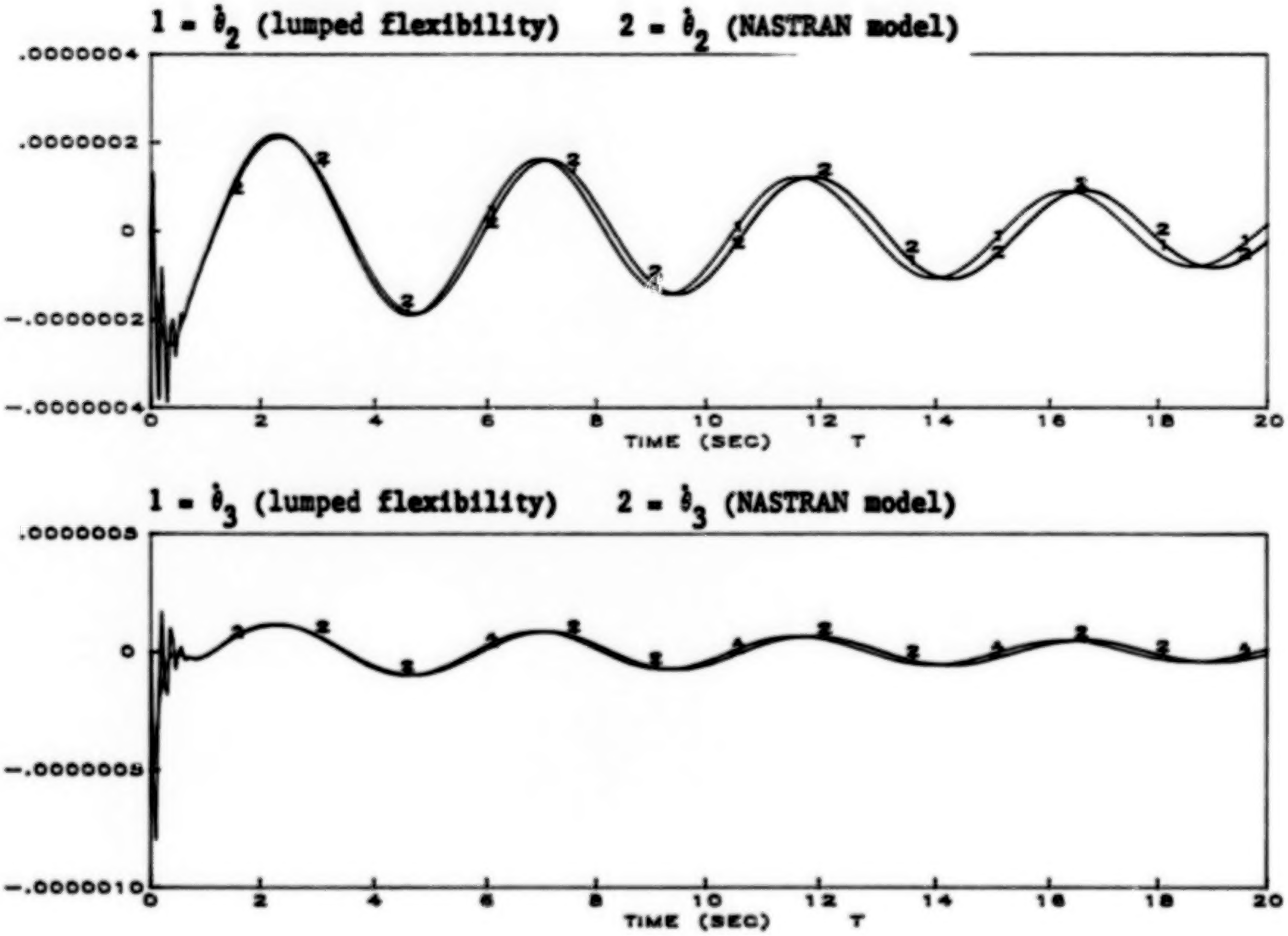


Figure 12.

BEST COPY AVAILABLE

Controlled Response
Lumped Flexibility Model vs NASTRAN Model (reduced order model)



1 = $\dot{\theta}_2$ (lumped flexibility)   2 = $\dot{\theta}_2$ (NASTRAN model)

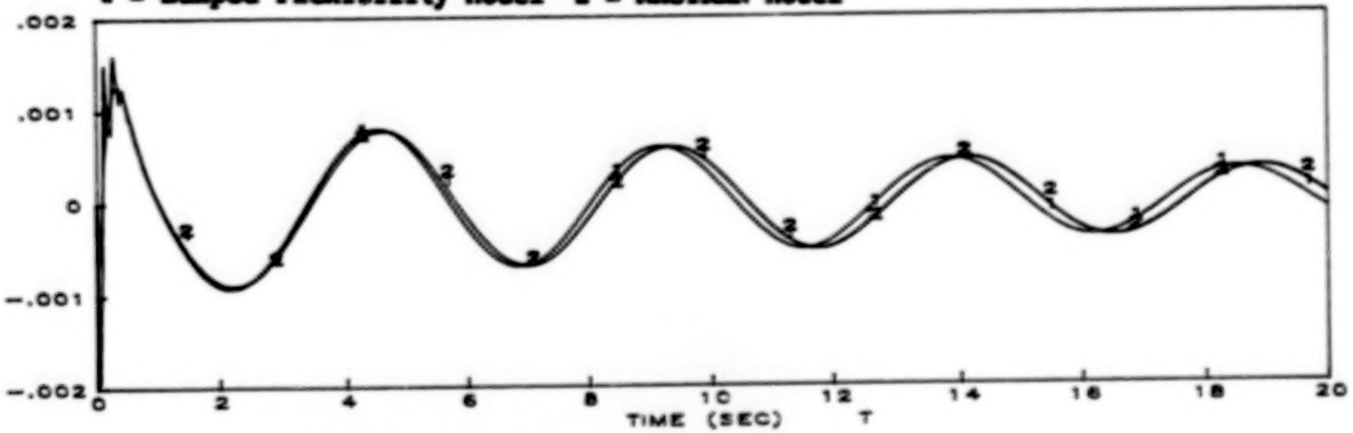
1 = $\dot{\theta}_3$ (lumped flexibility)   2 = $\dot{\theta}_3$ (NASTRAN model)

Figure 13.

BEST COPY
AVAILABLE

Actuator Inputs (Controller Outputs)
Lumped Flexibility vs NASTRAN Model

Torque Motor at Hinge 2
1 = Lumped Flexibility Model 2 = NASTRAN Model



Torque Motor at Hinge 3
1 = Lumped Flexibility Model 2 = NASTRAN Model

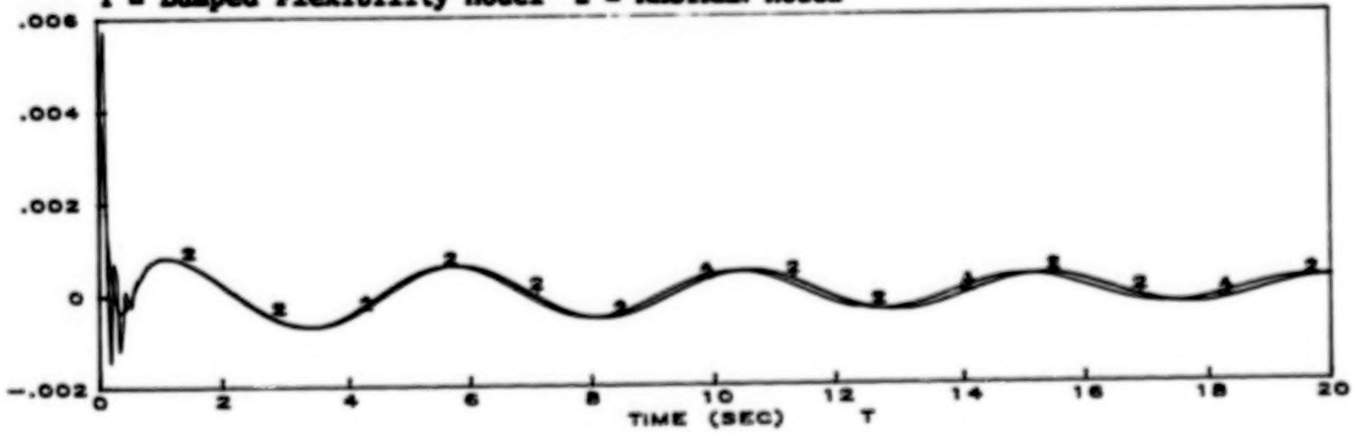


Figure 14.

158

BEST COPY AVAILABLE

ENHANCEMENTS TO THE IBM VERSION OF COSMIC/NASTRAN

by

W. Keith Brown
RPK Corporation
Hayes, Virginia

SUMMARY

Major improvements have been made to the IBM version of COSMIC/NASTRAN by RPK Corporation under contract to IBM Corporation. These improvements will become part of COSMIC's IBM version and will be available in the second quarter of 1989. The first improvement is the inclusion of code to take advantage of IBM's new Vector Facility (VF) on its 3090 machines. The remaining improvements are modifications that will benefit all users as a result of the extended addressing capability provided by the MVS/XA operating system. These improvements include the availability of an in-memory data base that potentially eliminates the need for I/O to the PRIxx disk files. Another improvement is the elimination of multiple load modules that have to be loaded for every link switch within NASTRAN. The last improvement allows for NASTRAN to execute above the 16 mega-byte line. This improvement allows for NASTRAN to have access to 2 giga-bytes of memory for open core and the in-memory data base.

INTRODUCTION

Very few changes have been made to the IBM version of COSMIC/NASTRAN in the last few years in order to take advantage of new hardware capabilities and new MVS/XA operating system features. One of IBM's new hardware capabilities is the Vector Facility that provides significant CPU time reductions for programs with vector operations (Reference 1). Use of IBM's new Vector Facility allows NASTRAN to solve larger problems in a much faster manner. Problems that spend a large amount of CPU time in symmetric decomposition, matrix multiplication, forward/backward substitution and eigenvalue analysis could greatly benefit from use of the Vector Facility. NASTRAN has been modified to take advantage of the Vector Facility in these areas.

With the release of MVS/XA, IBM allowed users to reference up to 2 giga-bytes of memory in a given job step. However, NASTRAN could only run under the 16 mega-byte line because of the assembly language code and the memory management design. Of the 16 mega-bytes, the most a user could get was about 8 mega-bytes because of the operating system. Although 8 mega-bytes was probably acceptable for open core, it was insufficient to contain any in-memory data files. NASTRAN has now been modified to allow it to execute above the 16 mega-byte line and this in turn allows access to a maximum of 2 giga-bytes of memory.

With access to 2 giga-bytes of memory, NASTRAN can now have the option of keeping DMAP files in memory. These files were previously written to the PRIxx files using BSAM I/O. This in-memory capability has been implemented and the implementation automatically allows for the use of external files when memory for the in-memory files has been exhausted. The end result of this feature is that job turnaround will be improved because of the reduction of disk I/O.

Lastly, NASTRAN was previously delivered as 16 load modules. One load module was always resident. The other 15 load modules were loaded into memory when needed but only one could reside in memory at a given time. If a new load module was needed, the current memory-resident load module would be deleted and the new load module loaded in its place. Therefore, as NASTRAN was processing the job, load modules would be loaded and then deleted to allow for other load modules. This resulted in lost time. Though the time that was lost was not appreciable, NASTRAN is now designed to eliminate this reloading procedure and all of NASTRAN now remains completely in memory.

EXECUTION TIME OPTIMIZATION CONSIDERATIONS

There are several considerations that a user should be aware of when trying to setup his NASTRAN run for optimal execution time. First, a user should be aware of the amount of open core that he may need. If open core is not large enough, users may experience severe execution time degradation due to possible spilling during decomposition or multiple passes on matrices used in matrix multiplication or forward-backward substitution. There is no single formula that may be used to determine the amount of open core needed for all cases; however, the following formulas provide some rough estimates.

Size = 6 * (larger of (number of degrees of freedom in the a-set)
or (number of degrees of freedom in the o-set))
* (number of output cases (e.g., number of loads or number of eigenvalues))

Size = .04 * (square of degrees of freedom in problem)

The above formulas are crude and should only be used as rough estimates. Users should check all symmetric decomposition messages to determine if any spill groups were required and also all matrix multiply-add (MPYAD) messages to determine if multiple passes were required. If either spill groups or multiple passes were required, then open core should be increased.

In addition to the amount of open core given to a problem, all problems will benefit from the use of the in-memory data base. The size of the in-memory data base is controlled by the REGION size given by the user, the memory to be returned to the operating system and the memory to be used for open core. The amount of memory to be used by the in-memory data base is computed as follows:

DB Size = (Region Size) - (Open Core Size) - (Operating System Memory)
- (NASTRAN Load Size)

(Note, the size of the NASTRAN load module is approximately 8000k bytes)

The in-memory data base capability eliminates much of the I/O performed by NASTRAN. This results in faster turnaround and faster execution. CPU savings will be of the order of about 5% for most problems.

USE OF IBM's VECTOR FACILITY

Efficiency improvements come also from the use of IBM's new Vector Facility. IBM's new Vector Facility allows programs with vector operations to use vector instructions for faster CPU execution (Reference 2). Inherent in the use of vector hardware is the length of the vector(s) to be processed. Due to the startup time and other associated overhead that goes with vector processing, vectors must be of a certain minimum length before CPU gains can be realized. In some cases where the vectors are very short, degradation can occur and longer CPU times can result. In general, vectors should have a length of 10 elements or more for real CPU gains to be realized. This is true for the majority of problems in NASTRAN.

NASTRAN has been optimized for vectorization in the following areas: symmetric decomposition, forward-backward substitution, eigenvalue analysis (Givens, Inverse Power and Feer) and matrix multiplication (Reference 3). The gains to be realized to a user are dependent upon the amount of the total CPU time that is spent in these areas. For most CPU intensive runs, analysis shows that these are the areas where most of the CPU time is used.

The modifications to NASTRAN took advantage of IBM's new Engineering and Scientific Subroutine Library (ESSL) (Reference 4). This library is a set of high performance mathematical subroutines that can be used by higher level languages such as Fortran. Use of the ESSL will ensure users of optimal performance in the future as the hardware characteristics of the Vector Facility may change.

Figure 1 shows improvements that were made on a statics, a normal modes and a frequency response problem using the optimized version of NASTRAN. The jobs were run on an IBM 3090E computer. A decrease of approximately 10% of the total CPU time could be realized if these problems were run on an IBM 3090S computer. The vector affinity column gives a measure of how much of the CPU time was spent in vector computations. Efficiency gains were realized from the vector code, the ability to have larger open core (eliminating multiple passes in matrix multiplication and forward-backward substitution) and the use of the in-memory data base (eliminating much of the I/O).

Depending on the problem characteristics, percentage improvements can vary greatly. Therefore, there is no hard and fast rule that users can use to show improvements. However, Figure 2 shows percentages that may be obtained during various matrix operations in NASTRAN. These figures are given only so the user can have some idea of expected improvements using the Vector Facility.

USE OF IBM's EXTENDED ARCHITECTURE (MVS/XA)

Prior to the release of IBM's MVS/XA operating system, the MVS operating system was based on 24-bit addressing. This meant that any given program could not address more than 2**16 bytes or 16 mega-bytes of memory. Of this 16 mega-bytes, approximately 8 mega-bytes were available to a program because of the operating system requirements. As software developments proceeded, especially in the area of graphics, and as user problems to be solved became larger, it became apparent that there was a need for programs to have access to more memory. With the coming of MVS/XA, IBM switched to 31-bit addressing and this allowed for 2**31 or 2 giga-bytes of memory to be available. Many of the IBM-supplied compilers were modified to take advantage of this and, most noticeably for NASTRAN, was the IBM VS Fortran compiler (Reference 5). Users could now write programs in Fortran and take full advantage of the 31-bit addressing capability (Reference 6).

The 16 mega-byte line that resulted from 24-bit addressing still exists in a limited sense for programs that have assembly language subroutines. Assembly language subroutines that perform I/O functions must execute in 24-bit addressing mode with the possible exception of subroutines that use execute channel program instructions (EXCPs) directly. Programs that have assembly language I/O must be designed to take this into account. The assembly code and the I/O buffers must reside below the 16 mega-byte line. In addition to the I/O considerations, sometimes assembly language codes have other 24-bit design dependent considerations that require that they execute in 24-bit addressing mode.

The previous IBM version of NASTRAN had a very small percentage of assembly language subroutines (Reference 7). These subroutines were required to allow for dynamic loading of NASTRAN load modules, to optimize computationally bound codes, to perform memory management and to provide efficient random access I/O. Because of these codes, NASTRAN required modifications to allow it to execute in 31-bit addressing mode. The IBM version of COSMIC/NASTRAN has now been redesigned to take advantage of 31-bit addressing. The new design resulted in two load modules. One load module is called NASTRAN and it executes above the 16 mega-byte line. This load module contains all of the analysis code and is by far the larger module. The other load module is called IO and it executes below the 16 mega-byte line. This module does all the non-Fortran I/O functions required by NASTRAN. Open core and the new in-memory data base reside above the 16 mega-byte line. Figure 3 shows this design.

New assembly language programs were created to allow for this design. The main program in the NASTRAN load module is the assembly language program NASTRAN. The main program in the IO load module is the assembly language program IO. The functions of the NASTRAN assembly language program are given below:

1. Reads and processes the job step parm. The format of the job parm has been changed and will be described below.

2. Initializes the Fortran run-time environment. The job parm is also passed by NASTRAN to Fortran during initialization to allow users the ability to take advantage of the new Fortran job parms that became available with Fortran Version 2. One such Fortran parameter that is recommended for use is the "NOXUFLOW" parameter that suppresses a program interrupt from a floating point underflow and allows for a hardware fixup instead of a software fixup.

3. Performs memory management. The REGION value as given on the job step EXEC card specifies the maximum amount of memory that NASTRAN can obtain. All memory that is available after the loading of the program is obtained. Memory is then released for the operating system use and the remaining memory is used by NASTRAN for open core and the in-memory data base.

4. Loads the IO load module and initializes the interface between the two. This allows for communication to all I/O subroutines that must reside below the 16 mega-byte line.

The IO program provides all of the non-Fortran I/O for the NASTRAN data files.

The job step parm has been modified for this new level of NASTRAN. The format is as follows:

```
// EXEC          PGM=NASTRAN,
// PARM='NOXUFLOW,OSMEM=200K,OCMEM=7000K,DBMEM=100000K'
```

The parameters are defined as follows:

NOXUFLOW	Parameter to Fortran to suspend underflow interceptions.
OSMEM	Amount of memory to free back to the operating system.
OCMEM	Amount of memory to allocate for open core.
DBMEM	Amount of memory to allocate for the in-memory data base. A non-zero value implies that memory is to be allocated for the in-memory data base. A zero value eliminates the use of the in-memory data base.

The last three parameters determine how memory is to be allocated during the run. NASTRAN first obtains all memory available based on the job step REGION value. It will then release back to the operating system the amount of memory as specified on the OSMEM parameter. The memory that remains will all be given to open core if DBMEM=0. Otherwise, the amount of memory as specified by the OCMEM parameter is designated for open core and all that remains is used for the in-memory data base. If the memory for the in-memory data base is greater than that specified on DBMEM parameter, the larger value is used. If there is a lesser amount of memory available than that specified on the DBMEM parameter, then the lesser amount is designated. If there is only sufficient memory for open core, then no memory is allocated for the in-memory data base regardless of the value of DBMEM.

NASTRAN prints a summary of the load addresses and the memory allocations at the beginning of the NASTRAN log file. A sample listing is shown in Figure 4.

IN-MEMORY DATA BASE

The in-memory data base will benefit users in job throughput and to a limited sense in CPU utilization. Job throughput will increase because of the elimination of I/O to disk. The CPU savings will be of the order of 5%. The in-memory data base is based on blocks of memory that are chained together. There is one chain for free space. The block sizes in the free chain will vary. There is also one chain for each DMAP file. Each DMAP file is a file in the in-memory data base. The size of the blocks allocated for the DMAP files is based upon the size specified for the NASTRAN GINO files. Blocks of memory are allocated for a DMAP file as the file is written and pointers are maintained as to the current block being processed. Memory that is available for allocation of the blocks is maintained by the free chain. As files are opened for write with rewind, the previously allocated memory data blocks to a DMAP file are released back into the free chain and a new DMAP file chain is established. Savings are realized because there is no I/O taking place and also there is no moving of data. All data being written into the DMAP file are written directly into the allocated memory block with no secondary transfer of data.

The in-memory data base is designed so that when there is insufficient memory for additional blocks of an existing DMAP file or for the creation of a new file, the external PRIxx, SECxx and TERxx files are used as spill. Users should be aware that the PRIxx, SECxx and TERxx DD cards are still needed, however the space allocations may need to be adjusted based on the amount of memory provided for the in-memory data base.

A directory of the in-memory data base is provided when DIAG 2 is turned on. The directory comes out after each DMAP module execution. Use of this DIAG is not recommended because of the large amounts of printout that it may generate. Users may opt to use the following technique to turn on DIAG 2 in order to get the in-memory data base directory at a specific point in the DMAP.

```
ALTER n $
PARAM //*DIAG*//2 $
ALTER n+1 $
PARAM //*DIAGOFF*//2 $
ENDALTER $
```

where n is a DMAP instruction number.

A sample printout of the in-memory data base is given in Figure 5. The unit number defines the unit number allocated to the DMAP file in the NASTRAN File Allocation Table (FIAT) and will be used as the "xx" value to determine which PRIxx file is to be used for spill. The name field gives the DMAP file name and the current number defines the current block at which the file is positioned. For files that are closed with rewind, this value will be zero. For files that are closed without rewind, this value will be the last block that was either read (file opened for read) or was written (file opened for write). The in-mem blocks value defines the number of blocks allocated in the in-memory data base to the DMAP file and the disk blocks value defines the number of blocks that could not be contained in memory and were written to the PRIxx external file. The trailer values are the matrix trailers associated with the DMAP file.

CONCLUDING REMARKS

The IBM version of COSMIC/NASTRAN has been enhanced to take advantage of IBM's Vector Facility and the extended addressing capability provided by IBM's MVS/XA operating system. The enhancements include modifications to subroutines that can take advantage of the Vector Facility, modifications to allow NASTRAN to execute above the 16 mega-byte line and modifications to allow for a new in-memory data base. When all of the above features are used, users will be pleased at the performance increase. In addition, these modifications open up the door for larger problems to be analyzed on IBM that were previously not practical because of CPU requirements and/or open core requirements.

TYPE OF PROBLEM	G-SET	A-SET	SCALAR CPU (Sec.)	VECTOR CPU (Sec.)	VECTOR AFFINITY (Sec.)	VECTOR CPU/ SCALAR CPU
Statics	28828	11426	1264.3	672.3	520.2	53%
Normal Modes	8946	225	634.9	382.6	205.8	60%
Freq. Resp.	3852	3614	584.7	390.2	281.0	67%

Figure 1. Samples of Improvements in CPU Utilitzation
(Problems executed on IBM 3090E computer)

MATRIX OPERATION	VECTOR CPU / SCALAR CPU
Symmetric Decomposition	45%
Forward-Backward Substitution	33%
Matrix Multiply-Add Method 1	72%
Matrix Multiply-Add Method 2	53%
Matrix Multiply-Add Method 3	20%

Figure 2. Improvements in Matrix Computations

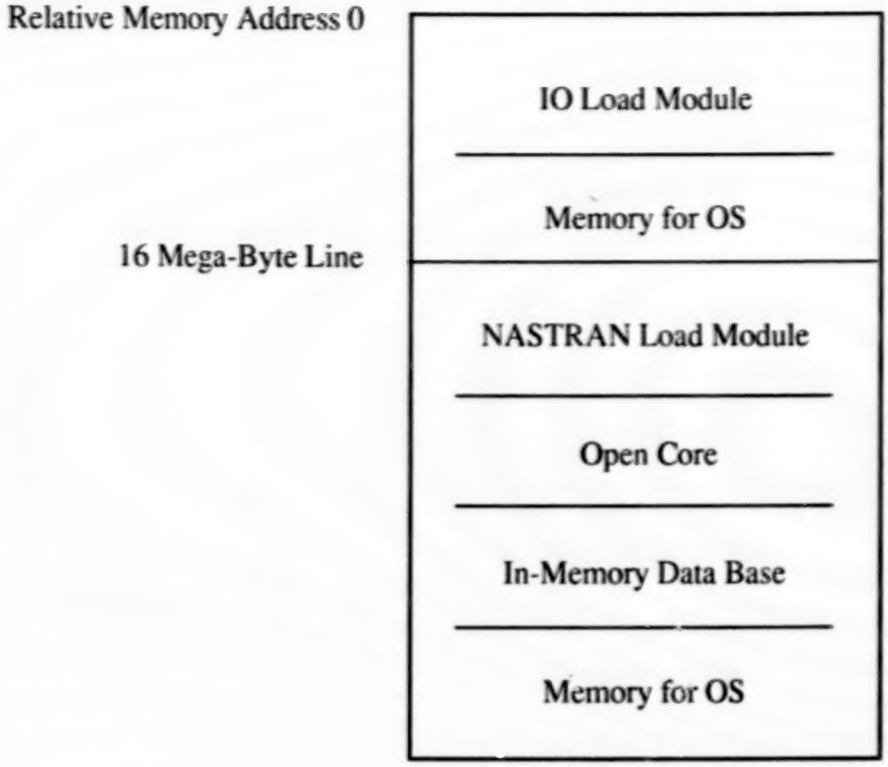

Relative Memory Address 0
IO Load Module
Memory for OS
16 Mega-Byte Line
NASTRAN Load Module
Open Core
In-Memory Data Base
Memory for OS

Figure 3. Memory Layout for IBM NASTRAN

```
ADDRESS OF BEGINNING OF NASTRAN LOAD MODULE = 04A009B0  HEX
ADDRESS OF END OF NASTRAN LOAD MODULE       = 0514D000  HEX
ADDRESS OF BEGINNING OF IO LOAD MODULE      = 8000CF30  HEX

ADDRESS OF MEMORY OBTAINED                  = 0514D000  HEX
LENGTH OF MEMORY OBTAINED                   = 0D17F000  HEX  219672576 DEC

MEMORY TO FREE FOR OS                       = 00032000  HEX     204800 DEC
MEMORY TO USE FOR OPENCORE                  = 005DC000  HEX    6144000 DEC
MEMORY TO USE FOR IN-MEM DATA BASE          = 0CB70FF8  HEX  213323768 DEC

ADDRESS OF BEGINNING OF OPENCORE            = 0514D000  HEX
ADDRESS OF END OF OPENCORE                  = 05729000  HEX
ADDRESS OF BEGINNING OF IN-MEM. DATA BASE   = 05729008  HEX
ADDRESS OF END OF IN-MEM. DATA BASE         = 12299FF8  HEX

(NOTE:  ALL UNITS ABOVE ARE IN BYTES)
```

Figure 4. Example of NASTRAN Printed Summary on Log File

```
                     MEMORY DATA BASE DIRECTORY
          MAXIMUM ENTRIES= 100     CURRENT ENTRIES=  33

 UNIT   NAME     CURRENT  IN-MEM   DISK               TRAILER
                 NUMBER   BLOCKS  BLOCKS

 1 20  PHIG         0       41      0       14  8946    2     2  8778  4906
 2  6  SIP          0        1      0        0     0    0     0     0     0
 3 22  SCRATCH1     0        1      0        1  8946    2     1   192   214
 4  9  CASECC       0        1      0        1     1  279     0     0     0
 5  4  BGPDP        0        2      0     1491     0    0     0     0     0
 6  5  SCRATCH2     0       39      0       14  8754    2     2  8394  4794
 7 14  QG           0        3      0        0     0    0     0     0     0
 8 12  KGGX         0      406      0     8946  8946    6     2   486    75
 9 10  MI           0        1      0       14    14    1     2    28 10000
10 19  MPTA         0        1      0    32768   128    0     0     0     0
11 18  SCRATCH6     0       26      0      225   225    6     2   450 10000
12 17  OEIGS        0        1      0       14   225    2     2   450 10000
13  7  GPL          0        2      0     1491     0    0     0     0     0
14 11  OPHIG        0        3      0        0  8512    0     0     0     0
15 13  CSTM         0        1      0     1491     2    0     0     0     0
16  8  EQEXIN       0        2      0     1491     0    0     0     0     0
17 15  BGPDT        0        2      0     1491     0    0     0     0     0
18 16  SIL          0        1      0     1491  8946    0     0     0     0
19 27  EST          0       10      0      560     0    0     6  1024     0
20 29  GPECT        0       33      0       82  1491    6   120     0     0
21 30  SCRATCH6     0       41      0       14  8946    2     2  8778  4906
22 32  MDICT        0        3      0        2     0    0     0     0     0
23 23  LAMA         0        1      0      225     0    0     0     0     0
24 21  SCRATCH3     0       29      0       14  4197    2     2  8394 10000
25 28  SCRATCH8     0       14      0      225   225    4     2   450  5022
26 33  GO           0      440      0      225  3972    2     2  7944 10000
27 34  USET         0        3      0        0  8946    0  2039     0     0
28 35  KFF          0      321      0     4197  4197    6     2   486   330
29 36  KFS          0       15      0     4557  4197    2     2   228    10
30 24  PHIA         0        2      0       14   225    2     2   450 10000
31 25  SCRATCH4     0        2      0       14   192    2     2   384 10000
32 26  SCRATCH5     0        2      0       14   225    2     2   450 10000
33 31  USETD        0        3      0        0     0    0     0     0     0

TOTAL IN-MEMORY BLOCKS =   1453
TOTAL DISK BLOCKS      =      0

TOTAL FREE SPACE IN WORDS             =  48508896
NUMBER OF BLOCKS IN FREE SPACE CHAIN  =      2585
```

Figure 5. Example of In-Memory Data Base Directory

REFERENCES

1. IBM System/370 Vector Operations, SA22-7125-2, Third Edition, August, 1987.

2. IBM Designing and Writing Fortran Programs for Vector and Parallel Processing, SC23-0337-00, First Edition, November, 1986.

3. The NASTRAN Theoretical Manual, NASA SP-221(06), January, 1981.

4. IBM Engineering and Scientific Subroutine Library Guide and Reference Release 3, SC23-0184-3, Fourth Edition, November, 1988.

5. IBM VS Fortran Version 2 Language and Library Reference Release 3, SC26-4221-3, Fourth Edition, March, 1988.

6. IBM Fortran Version 2 Programming Guide Release 3, SC26-4222-3, Fourth Edition, March, 1988.

7. The NASTRAN Programmer's Manual, NASA SP-223(05), Level 17.5, December, 1978.

A POWERFUL ENHANCEMENT TO THE DMAP ALTER CAPABILITY

by

P. R. Pamidi
RPK Corporation
Columbia, Maryland

SUMMARY

A powerful enhancement to the DMAP alter capability has been developed by RPK Corporation and is available on all RPK-supported versions of COSMIC/NASTRAN. This enhancement involves the addition of two new alter control cards, called INSERT and DELETE, to the Executive Control Deck. These cards allow for DMAP alters to be made by referencing DMAP statements by their module names rather than by their statement numbers in the rigid format DMAP sequence. This allows for increased user convenience and flexibility and makes alters more meaningful to the user. In addition, DMAP alter packages employing the new alter control cards will be much less susceptible to future changes in rigid format DMAPs than alter packages employing the standard ALTER control cards. The usage of the new cards is illustrated by examples.

INTRODUCTION

The most general way of using NASTRAN is by means of a user-written Direct Matrix Abstraction Program (DMAP). However, in order to relieve the user of the burden of constructing DMAP sequences for each of his analyses, standard DMAP sequences, called rigid formats, are provided with NASTRAN to handle different types of analyses.

It is often desirable for the user to make changes to the DMAP sequences in the rigid formats. This can be accomplished by using the DMAP alter capability (see Reference 1). Typical situations that may call for using DMAP alters are to schedule an exit prior to completion, to request additional intermediate output, to schedule diagnostic printing of tables and/or matrices and to modify the standard solution sequences by the addition and/or deletion of functional modules.

DESCRIPTION OF THE STANDARD ALTER FEATURE

DMAP alters to the rigid formats are accomplished by means of ALTER control cards in the Executive Control Deck (Reference 1). ALTER control cards are of two types.

An ALTER control card of the form

```
ALTER n $
```

indicates that DMAP instructions following this card are to be inserted after DMAP instruction number n in the rigid format under consideration.

An ALTER control card of the form

ALTER n1,n2 $ (n1 <= n2)

indicates that DMAP instructions in the range n1 through n2 (inclusive) in the rigid format are to be deleted and replaced by any DMAP instructions that may follow this card.

The ALTER control cards serve a very useful purpose. However, the usage of these cards has the following two distinct disadvantages:

- The ALTER control cards refer to DMAP statements by their numbers in the rigid format DMAP sequence. This does not give a "feel" for the DMAP changes as the numbers do not have any particular significance to the user. In other words, the ALTER control cards are by design really more programmer-oriented than user-oriented.

- Because the ALTER control cards refer to DMAP statements by numbers, they are very susceptible to changes in rigid formats from one release to a subsequent one. Thus, even minor changes in a rigid format, particularly in the earlier portion of the DMAP sequence, may require wholesale revamping of the ALTER cards in an alter package.

DESCRIPTION OF THE ENHANCED ALTER FEATURES

In order to overcome the above shortcomings, RPK has developed a very attractive enhancement to the DMAP alter capability. This enhancement involves the addition of two new alter control cards, called INSERT and DELETE, for use in the Executive Control Deck. This feature is available on all RPK-supported versions of COSMIC/NASTRAN, beginning with the 1988 release.

Detailed descriptions of the INSERT and DELETE cards are given in Appendix A. An updated description of the ALTER card that takes into account the existence of the INSERT and DELETE cards is also given in that appendix.

The INSERT control card identifies a specific module in the rigid format DMAP sequence after which DMAP instructions following the INSERT card are to be inserted. The DELETE control card identifies a specific module (or a range of modules) in the rigid format DMAP sequence which is (or are) to be deleted and replaced by any DMAP instructions that may follow the DELETE card.

The INSERT control card is specified as follows:

INSERT specmod $

where specmod has the following general form:

nommod [(r)] [, n]

The various terms in the above specification have the following meanings and connotations:

nommod	is the *nominal* module (alphanumeric value, no default). This must be a valid name of a module in the rigid format DMAP sequence. (It must be recognized in this context that every DMAP instruction or DMAP statement is a module with a specific name.)
r	is the occurrence flag (integer > 0, default $= 1$). The r^{th} occurrence of the nominal module in the rigid format DMAP sequence (counting from the beginning of the DMAP sequence) defines the *reference* module. The default value of 1 for the occurrence flag implies that the reference module is the first occurrence of the nominal module in the rigid format DMAP sequence.
n	is the offset flag (integer, default $= 0$). The DMAP module that is offset from the reference module by n DMAP statements in the rigid format DMAP sequence defines the *specified* module. Depending upon the sign of the offset flag n, the specified module may follow (n positive) or precede (n negative) the reference module in the rigid format DMAP sequence. The default value of 0 for the offset flag implies that the reference module is the specified module.
specmod	is the module defined as per the above scheme after which DMAP statements following the INSERT card are to be inserted.

The DELETE control card is specified as follows:

$$\text{DELETE specmod}_1\ [,\ \text{specmod}_2\ ]\ \$$$

where specmod_i has the following general form:

$$\text{nommod}_i\ [(r_i)]\ [,\ n_i\ ]$$

The various terms in the above specification have the same meanings and connotations as in the case of the INSERT control card.

If only specmod_1 is specified on a DELETE card, it identifies a single specified module that is to be deleted and replaced by any DMAP statements that may follow the DELETE card. If both specmod_1 and specmod_2 are specified, they identify a range of specified modules that are to be deleted and replaced by any DMAP statements that may follow the DELETE card.

USAGE OF THE ENHANCED ALTER FEATURES

The new INSERT and DELETE cards described above and the existing ALTER card together form a triad of alter control cards available to the user on all RPK-supported versions of COSMIC/NASTRAN. When using these cards, the most important requirement that must be satisfied is the one that has always existed with the usage of the standard ALTER control cards, namely, that the DMAP statements (or modules) that are referenced on the ALTER, INSERT and DELETE control cards in an alter package (either explicitly or implicitly, when a range is specified) must be referenced in ascending order of their occurrence in the rigid format DMAP.

The new INSERT and DELETE cards can be used in conjunction with standard ALTER control cards and any combination of the three control cards is acceptable. As a corollary, RPK-supported versions of COSMIC/NASTRAN also support alter packages containing only ALTER control cards. This ensures compatibility with standard versions of COSMIC/NASTRAN.

Table 1 lists several examples of the usage of alter control cards on RPK-supported versions of COSMIC/NASTRAN. For each example, the table shows an alter using standard ALTER control cards and indicates suggested usages by which the same alter can be accomplished by employing equivalent INSERT or DELETE control cards. (All of the examples in the table refer to the DMAP sequence of Rigid Format 3 - Displacement Approach, Release 1988, that is given in Appendix B.)

RPK encourages the users of its versions of COSMIC/NASTRAN to use the new alter control cards. In order to demonstrate their usage, RPK has modified the data for all NASTRAN Demonstration Problems that contain ALTER cards by commenting out all such cards and replacing them by equivalent INSERT and/or DELETE cards. This is reflected in the data and the output of the NASTRAN Demonstration Problems that are delivered to RPK's clients.

ADVANTAGES OF THE ENHANCED ALTER FEATURES

The new alter control cards have several distinct advantages over the standard ALTER control card. Some of these are obvious from the examples in Table 1. These advantages are discussed in detail below.

1. Increased User Friendliness and Convenience

Unlike standard ALTER control cards, which refer to DMAP instructions by their statement numbers in the rigid format DMAP, the new INSERT and DELETE control cards refer to DMAP statements by their module names. This is certainly more user friendly and convenient as DMAP module names are clearly more meaningful to the user than DMAP statement numbers. The user thus has a better "feel" for the alters.

2. Increased Flexibility

The general manner in which the specified module is identified on the INSERT and DELETE control cards gives tremendous flexibility to the user.

Using ALTER cards, a given alter can be accomplished only by a very specific and unique ALTER card. However, by using INSERT (or DELETE) cards, the same alter can be accomplished in several apparently different, but equivalent, ways. The user thus has a choice of ways in which he can specify a given alter.

The above point can be best illustrated by an example. Consider Example 2 in Table 1 which indicates that alters are to be made by inserting new DMAP statements after DMAP statement no. 69 (the PARAM module just before the READ module) in the DMAP.

By using ALTER control cards, the above alter can be accomplished only by using the following very specific and unique alter:

```
ALTER 69 $
```

However, by using INSERT control cards, the above alter can be accomplished in many different ways. The following are some ways of achieving this (the first two alters given below are shown in Table 1 for this example):

```
INSERT DPD,2 $
INSERT READ,-1 $
INSERT RBMG4,4 $
INSERT SDR1,-5 $
INSERT BEGIN,68 $
INSERT END,-30 $
```

All of the above INSERT cards (the last two INSERTs shown above are admittedly extreme examples), though different in appearance, are all equivalent since they identify the same specified module, namely, the PARAM module just before the READ module in the DMAP. They differ from one another in that each of them employs a different reference module in conjunction with a correspondingly different offset flag.

In a similar manner, if alters involve the deletion of DMAP modules, DELETE control cards can be used to accomplish it in more than one way.

Assume that the number of DMAP statements in a rigid format DMAP sequence is m. Then, by using INSERT control cards, a given alter of the form

```
ALTER n $     or    ALTER n,n $
```

can be accomplished in m different, but equivalent, ways by selecting each of the m DMAP modules in the rigid format as a reference module with an appropriate offset flag.

In a similar manner, a given alter of the form

```
ALTER n1,n2 $   (n1 # n2)
```

can be accomplished in m^2 different ways since each of n1 and n2 can be specified in m different, but equivalent, ways.

3. Reduced Susceptibility to Future Changes in Rigid Format DMAPs

Because the new alter control cards refer to DMAP statements by their module names, alter packages that contain these new cards will be much less susceptible to future changes in rigid formats than if standard ALTER cards were used.

Consider, for instance, Example 1 in Table 1. This involves the insertion of new DMAP statements after the SDR2 module. (This is the normal alter that is used to obtain NASTRAN output for subsequent interface with post-processing programs like PATRAN.)

The only way of accomplishing the above alter by using standard ALTER control cards is to use the following alter :

ALTER 79 $

The above alter will no longer be valid if future changes to the rigid format involve additions or deletions to the DMAP ahead of the SDR2 module. In that case, the new DMAP statement number for the SDR2 module must be used in the above alter.

By using INSERT control cards, the above alter can be accomplished by the following alter:

INSERT SDR2 $

Because the above alter refers to the DMAP module by name, it will be unaffected by any future additions or deletions to the DMAP.

CONCLUDING REMARKS

This paper has described a powerful enhancement to the DMAP alter capability that has been developed by RPK Corporation and that is available on all RPK-supported versions of COSMIC/NASTRAN. This enhancement involves the addition of two new alter control cards, called INSERT and DELETE, to the Executive Control Deck. These cards allow for DMAP alters to be made by referencing DMAP statements by their module names rather than by their statement numbers in the rigid format DMAP sequence. This allows for increased user convenience and flexibility and makes alters more meaningful to the user. In addition, DMAP alter packages employing the new alter control cards will be much less susceptible to future changes in rigid format DMAPs than alter packages employing the standard ALTER control cards. The usage of the new cards is illustrated by examples.

REFERENCE

1. The NASTRAN User's Manual, NASA SP-222(08), June 1986.

Table 1. Examples on the Usage of Alter Control Cards

(see Note 1 below)

Example no.	Alters using ALTER cards	Equivalent alters using INSERT or DELETE cards (see Note 2 below)
1	ALTER 79 $	INSERT SDR2 $
2	ALTER 69 $	INSERT DPD,2 INSERT READ,-1 $
3	ALTER 31 $	INSERT EMA(2) $
4	ALTER 30 $	INSERT EMA,2 $ INSERT EMA(2),-1 $
5	ALTER 82,82 $	DELETE SCAN $
6	ALTER 2,3 $	DELETE PRECHK,FILE $
7	ALTER 32,35 $	DELETE GPWG,-1,GPWG,2 $
8	ALTER 84,87 $	DELETE PLOT(2),-1,PLOT(2),2 $

Notes:

1. All of the alters given above refer to the DMAP sequence of Rigid Format 3 - Displacement Approach, Release 1988, given in Appendix B.

2. The equivalent alters using INSERT or DELETE cards shown above are only suggested usages. As explained in the paper, alters using INSERT or DELETE control cards are not unique and can be accomplished in more than one way.

APPENDIX A

Description of Alter Control Cards

Executive Control Card ALTER - Rigid Format DMAP Sequence Alteration Request

Description: Requests the Direct Matrix Abstraction Program (DMAP) sequence of a rigid format to be changed by additions, deletions or substitutions.

Format and Examples:

ALTER{K1 [, K2]}$

ALTER 22 $

ALTER 5,5 $

ALTER 38,45 $

ALTER 25,19 $

Option	
K1 only	DMAP statement number (Integer > 0) after which DMAP instructions following the ALTER card are to be inserted
K1 and K2	DMAP statement numbers (Integer > 0) identifying a single DMAP statement or a range of DMAP statements to be deleted and replaced by any DMAP instructions that may follow the ALTER card. See Remark 5.

Remarks:

1. See the descriptions of the INSERT and DELETE cards for alternate ways of specifying DMAP sequence alteration requests.
2. The DMAP statements referenced on ALTER, INSERT and DELETE cards (either explicitly or implicitly, when a range is specified) must be referenced in ascending order of their occurrence in the rigid format DMAP.
3. See Volume 2, Sections 2, 3 and 4 for the listings of all rigid format DMAP sequences.
4. See Volume 2, Section 1.1.5 for the manner in which DMAP alters are handled in restarts.
5. If both K1 and K2 are specified and K1 ≠ K2, a range of DMAP statements is implied and either of them can be less than the other. If K1 = K2, a single DMAP statement is implied.

BEST COPY
AVAILABLE

Executive Control Card DELETE - Rigid Format DMAP Sequence Alteration Request

Description: Requests the Direct Matrix Abstraction Program (DMAP) sequence of a rigid format to be changed by deletions or substitutions.

Format and Examples:

DELETE specmod_1 [, specmod_2] $

where specmod_i has the following general form:

nommod_i [(r_i)] [, n_i]

```
DELETE SSG1 $

DELETE EMA(2) $

DELETE READ,1 $

DELETE SDR2(2),-1 $

DELETE SSG3,REPT $

DELETE GP2,GP3,-1 $

DELETE SMA3,1,TA1,-1 $

DELETE REPT,2,REPT,3 $
```

Option	
nommod_i	Nominal module (Alphanumeric value, no default). See Remark 5.
r_i	Occurrence flag (Integer > 0, default = 1). The r_i^{th} occurrence of the nominal module in the rigid format DMAP sequence (counting from the beginning of the DMAP sequence) defines the reference module. See Remark 6.
n_i	Offset flag (Integer, default = 0). The DMAP module that is offset from the reference module by n_i DMAP statements in the rigid format DMAP sequence defines the specified module. See Remark 7.
specmod_1 only	Specified module defined as per the above scheme that is to be deleted and replaced by any DMAP instructions that may follow the DELETE card
specmod_1 and specmod_2	Range of specified modules defined as per the above scheme that are to be deleted and replaced by any DMAP instructions that may follow the DELETE card. See Remark 8.

Remarks:

1. See the description of the ALTER card for an alternate way of specifying DMAP sequence deletions and substitutions.
2. The DMAP statements referenced on ALTER, INSERT and DELETE cards (either explicitly or implicitly, when a range is specified) must be referenced in ascending order of their occurrence in the rigid format DMAP.

3. See Volume 2, Sections 2, 3 and 4 for the listings of all rigid format DMAP sequences.

4. See Volume 2, Section 1.1.5 for the manner in which DMAP alters are handled in restarts.

5. The nominal module $nommod_1$ must be a valid name of a DMAP module in the rigid format DMAP sequence.

6. The default value of 1 for the occurrence flag r_1 implies that the reference module is the first occurrence of the nominal module in the rigid format DMAP sequence.

7. The value of the offset flag n_1 may be positive, negative or 0. A positive value means that the specified module follows the reference module by n_1 DMAP statements in the rigid format DMAP sequence. A negative value indicates that the specified module precedes the referance module by n_1 DMAP statements in the DMAP sequence. A value of 0 (the default) implies that the reference module is the specified module.

8. If both $specmod_1$ and $specmod_2$ are specified, it implies a range of DMAP statements and either of them can precede the other in the rigid format DMAP sequence.

Executive Control Card INSERT - Rigid Format DMAP Sequence Alteration Request

Description: Requests the Direct Matrix Abstraction Program (DMAP) sequence of a rigid format to be changed by additions.

Format and Examples:

```
INSERT  specmod $

        where specmod has the following general form:

                        nommod [(r)] [, n]

INSERT GP4 $

INSERT EMA(2) $

INSERT READ,1 $

INSERT SDR2(2),-1 $
```

Option

nommod	Nominal module (Alphanumeric value, no default). See Remark 5.
r	Occurrence flag (Integer > 0, default = 1). The r^{th} occurrence of the nominal module in the rigid format DMAP sequence (counting from the beginning of the DMAP sequence) defines the reference module. See Remark 6.
n	Offset flag (Integer, default = 0). The DMAP module that is offset from the reference module by n DMAP statements in the rigid format DMAP sequence defines the specified module. See Remark 7.
specmod	Specified module defined as per the above scheme after which DMAP statements following the INSERT card are to be inserted.

Remarks:

1. See the description of the ALTER card for an alternate way of specifying DMAP sequence additions.
2. The DMAP statements referenced on ALTER, INSERT and DELETE cards (either explicitly or implicitly, when a range is specified) must be referenced in ascending order of their occurrence in the rigid format DMAP.
3. See Volume 2, Sections 2, 3 and 4 for the listings of all rigid format DMAP sequences.
4. See Volume 2, Section 1.1.5 for the manner in which DMAP alters are handled in restarts.
5. The nominal module nommod must be a valid name of a DMAP module in the rigid format DMAP sequence.
6. The default value of 1 for the occurrence flag r implies that the reference module is the first occurrence of the nominal module in the rigid format DMAP sequence.

7. The value of the offset flag n may be positive, negative or 0. A positive value means that the specified module follows the reference module by n DMAP statements in the rigid format DMAP sequence. A negative value indicates that the specified module precedes the reference module by n DMAP statements in the DMAP sequence. A value of 0 (the default) implies that the reference module is the specified module.

APPENDIX B

DMAP Listing of Rigid Format 3 - Displacement Approach, Release 1988

```
                    LEVEL 2.0 NASTRAN DMAP COMPILER - SOURCE LISTING

OPTIONS IN EFFECT  GO    ERR=2  LIST   NODECK  NOREF  NOOSCAR
------------------

      1  BEGIN      DISP 03 - NORMAL MODES ANALYSIS - APR. 1988 $

      2  PRECHK     ALL$

      3  FILE       LAMA=APPEND/PHIA=APPEND $

      4  PARAM      //*MPY*/CARDNO/0/0 $

      5  GP1        GEOM1,GEOM2,/GPL,EQEXIN,GPDT,CSTM,BGPDT,SIL/S,N,LUSET/
                    NOGPDT/ALWAYS=-1 $

      6  PLTTRAN    BGPDT,SIL/BGPDP,SIP/LUSET/S,N,LUSEP $

      7  GP2        GEOM2,EQEXIN/ECT $

      8  PARAML     PCDB//*PRES*////JUMPPLOT $

      9  PURGE      PLTSETX,PLTPAR,GPSETS,ELSETS/JUMPPLOT $

     10  COND       P1,JUMPPLOT $

     11  PLTSET     PCDB,EQEXIN,ECT/PLTSETX,PLTPAR,GPSETS,ELSETS/S,N,NSIL/
                    S,N,JUMPPLOT $

     12  PRTMSG     PLTSETX// $

     13  PARAM      //*MPY*/PLTFLG/1/1 $

     14  PARAM      //*MPY*/PFILE/0/0 $

     15  COND       P1,JUMPPLOT $

     16  PLOT       PLTPAR,GPSETS,ELSETS,CASECC,BGPDT,EQEXIN,SIL,,ECT,,,,/PLOTX1/
                    NSIL/LUSET/S,N,JUMPPLOT/S,N,PLTFLG/S,N,PFILE  $

     17  PRTMSG     PLOTX1//$

     18  LABEL      P1 $

     19  GP3        GEOM3,EQEXIN,GEOM2/,GPTT/NOGRAV $

     20  TA1        ECT,EPT,BGPDT,SIL,GPTT,CSTM,MPT/EST,GEI,GPECT,,,MPTX,PCOMPS,
                    EPTX/LUSET/S,N,NOSIMP/1/S,N,NOGENL/GENEL/S,N,COMPS $

     21  EQUIV      MPTX,MPT/COMPS/EPTX,EPT/COMPS $

     22  COND       ERROR4,NOSIMP $

     23  PARAM      //*ADD*/NOKGGX/1/0 $

     24  PARAM      //*ADD*/NOMGG/1/0 $
```

```
25  EMG       EST,CSTM,MPT,DIT,GEOM2,/KELM,KDICT,MELM,MDICT,,,/S,N,NOKGGX/
              S,N,NOMGG////C,Y,COUPMASS/C,Y,CPBAR/
              C,Y,CPROD/C,Y,CPQUAD1/C,Y,CPQUAD2/C,Y,CPTRIA1/C,Y,CPTRIA2/
              C,Y,CPTUBE/C,Y,CPQDPLT/C,Y,CPTRPLT/C,Y,CPTRBSC/
              C,Y,VOLUME/C,Y,SURFACE $

26  PURGE     KGGX/NOKGGX $

27  COND      JMPKGG,NOKGGX $

28  EMA       GPECT,KDICT,KELM/KGGX $

29  LABEL     JMPKGG $

30  COND      ERROR1,NOMGG $

31  EMA       GPECT,MDICT,MELM/MGG/-1/C,Y,WTMASS=1.0 $

32  COND      LGPWG,GRDPNT $

33  GPWG      BGPDP,CSTM,EQEXIN,MGG/OGPWG/V,Y,GRDPNT=-1/C,Y,WTMASS $

34  OFP       OGPWG,,,,,//S,N,CARDNO $

35  LABEL     LGPWG $

36  EQUIV     KGGX,KGG/NOGENL $

37  COND      LBL11,NOGENL $

38  SMA3      GEI,KGGX/KGG/LUSET/NOGENL/NOSIMP $

39  LABEL     LBL11 $

40  GPSTGEN   KGG,SIL/GPST $

41  PARAM     //*MPY*/NSKIP/0/0 $

42  GP4       CASECC,GEOM4,EQEXIN,GPDT,BGPDT,CSTM,GPST/RG,YS,USET,
              ASET,OGPST/LUSET/S,N,MPCF1/S,N,MPCF2/S,N,SINGLE/S,N,OMIT/
              S,N,REACT/S,N,NSKIP/S,N,REPEAT/S,N,NOSET/S,N,NOL/S,N,NOA/
              C,Y,ASETOUT/C,Y,AUTOSPC $

43  OFP       OGPST,,,,,//S,N,CARDNO $

44  COND      ERROR3,NOL $

45  PURGE     KRR,KLR,DM,MLR,MR/REACT/GM/MPCF1/GO/OMIT/KFS/SINGLE/QG/NOSET $

46  EQUIV     KGG,KNN/MPCF1/MGG,MNN/MPCF1 $

47  COND      LBL2,MPCF1 $

48  MCE1      USET,RG/GM $

49  MCE2      USET,GM,KGG,MGG,,/KNN,MNN,, $
```

```
50  LABEL     LBL2 $

51  EQUIV     KNN,KFF/SINGLE/MNN,MFF/SINGLE $

52  COND      LBL3,SINGLE $

53  SCE1      USET,KNN,MNN,,/KFF,KFS,,MFF,, $

54  LABEL     LBL3  $

55  EQUIV     KFF,KAA/OMIT $

56  EQUIV     MFF,MAA/OMIT $

57  COND      LBL5,OMIT $

58  SMP1      USET,KFF,,,/GO,KAA,KOO,LOO,,,,, $

59  SMP2      USET,GO,MFF/MAA $

60  LABEL     LBL5 $

61  COND      LBL6,REACT $

62  RBMG1     USET,KAA,MAA/KLL,KLR,KRR,MLL,MLR,MRR $

63  RBMG2     KLL/LLL $

64  RBMG3     LLL,KLR,KRR/DM $

65  RBMG4     DM,MLL,MLR,MRR/MR $

66  LABEL     LBL6 $

67  DPD       DYNAMICS,GPL,SIL,USET/GPLD,SILD,USETD,,,,,,,EED,EQDYN/
              LUSET/LUSETD/NOTFL/NODLT/NOPSDL/NOFRL/
              NONLFT/NOTRL/S,N,NOEED//NOUE $

68  COND      ERROR2,NOEED $

69  PARAM     //*MPY*/NEIGV/1/-1 $

70  READ      KAA,MAA,MR,DM,EED,USET,CASECC/LAMA,PHIA,MI,OEIGS/*MODES*/
              S,N,NEIGV $

71  OFP       OEIGS,,,,,//S,N,CARDNO $

72  COND      FINIS,NEIGV $

73  OFP       LAMA,,,,,//S,N,CARDNO $

74  SDR1      USET,,PHIA,,,GO,GM,,KFS,,/PHIG,,QG/1/*REIG* $

75  COND      NOMPCF,GRDEQ $

76  EQMC      CASECC,EQEXIN,GPL,BGPDT,SIL,USET,KGG,GM,PHIG,LAMA,QG,CSTM/
              OQM1/V,Y,OPT=0/V,Y,GRDEQ/-1 $
```

```
77  OFP         OQM1,,,,,//S,N,CARDNO $

78  LABEL       NOMPCF $

79  SDR2        CASECC,CSTM,MPT,DIT,EQEXIN,SIL,,,BGPDP,LAMA,QG,PHIG,EST,,,
                PCOMPS/,OQG1,OPHIG,OES1,OEF1,PPHIG,OES1L,OEF1L/
                *REIG*////COMPS $

80  OFP         OPHIG,OQG1,OEF1,OES1,,//S,N,CARDNO $

81  OFP         OEF1L,OES1L,,,,//S,N,CARDNO $

82  SCAN        CASECC,OES1,OEF1/OESF1/*RF* $

83  OFP         OESF1,,,,,//S,N,CARDNO $

84  COND        P2,JUMPPLOT $

85  PLOT        PLTPAR,GPSETS,ELSETS,CASECC,BGPDT,EQEXIN,SIP,,PPHIG,GPECT,OES1,
                OES1L,/PLOTX2/NSIL/LUSEP/JUMPPLOT/PLTFLG/S,N,PFILE $

86  PRTMSG      PLOTX2// $

87  LABEL       P2 $

88  JUMP        FINIS $

89  LABEL       ERROR1 $

90  PRTPARM     //-1/*MODES* $

91  LABEL       ERROR2 $

92  PRTPARM     //-2/*MODES* $

93  LABEL       ERROR3 $

94  PRTPARM     //-3/*MODES* $

95  LABEL       ERROR4 $

96  PRTPARM     //-4/*MODES* $

97  LABEL       FINIS $

98  PURGE       DUMMY/ALWAYS $

99  END         $
```

A NASTRAN DMAP ALTER FOR LINEAR BUCKLING ANALYSIS UNDER DYNAMIC LOADING

Robert A. Aiello and Joseph E. Grady
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

SUMMARY

A unique modification to the NASTRAN solution sequence for transient analysis with direct time integration (COSMIC NASTRAN rigid format 9) has been developed and incorporated into a DMAP alter. This DMAP alter calculates the buckling stability of a dynamically loaded structure, and is used to predict the onset of structural buckling under stress-wave loading conditions. The modified solution sequence incorporates the linear buckling analysis capability (rigid format 5) of NASTRAN into the existing Transient solution rigid format in such a way as to provide a time dependent eigensolution which is used to assess the buckling stability of the structure as it responds to the impulsive load. As a demonstration of the validity of this modified solution procedure, the dynamic buckling of a prismatic bar subjected to an impulsive longitudinal compression is analyzed and compared to the known theoretical solution. In addition, a dynamic buckling analysis is performed for the analytically less tractable problem of the localized dynamic buckling of an initially flawed composite laminate under transverse impact loading. The addition of this DMAP alter to the transient solution sequence in NASTRAN facilitates the computational prediction of both the time at which the onset of dynamic buckling occurs in an impulsively loaded structure, and the dynamic buckling mode shapes of that structure.

INTRODUCTION

Composite laminates that are subjected to static, dynamic, or fatigue loading are known to undergo delamination, or debonding, between the laminated plies of which they are composed. Delamination causes a significant loss stiffness and strength, and can considerably reduce the structural integrity of a laminate. Once this damage has occurred, a compressive stress near the delamination can induce local buckling of the delaminated plies. This buckling may then cause further extension of the delamination and progressive weakening of the laminate. In lieu of actual experimental testing, the ability to computationally predict the onset of delamination buckling is necessary for evaluating the durability of many composite structures.

The delamination buckling phenomenon has been observed experimentally under both static and fatigue loading conditions (Refs. 1 to 4), and several analytical and numerical methods have been proposed (Refs. 5 and 6) to model this damage mechanism. Finite-element approaches (Refs. 7 to 9) have been used as the basis for these analyses, but no comparable numerical methods exist to analyze delimination buckling which occurs as a result of an impulsively applied load. That is the topic of this paper.

Experimental observations of dynamic delamination buckling in transversely impacted laminates were reported earlier (Refs. 10 to 12), using high-speed photography and simultaneous strain measurements of transversely impacted laminates. A related numerical analysis (Ref. 10) indicated that the buckling behavior must be accounted for in the computational model in order to accurately assess the damage tolerance capability of the laminate. This motivated the present development of a NASTRAN DMAP alter analysis procedure that can be used to computationally predict the onset of buckling instability under transient stress-wave loading.

The objectives of this paper are, therefore, (1) to outline the dynamic buckling analysis computational procedure and its implementation into the DMAP alter sequence (2) demonstrate the validity of the dynamic buckling analysis procedure by analyzing a simple one-dimensional example problem with a known solution, and (3) apply the dynamic buckling analysis to the analytically less tractable problem of the localized dynamic buckling of an initially flawed composite laminate under transverse impact loading.

The NASTRAN transient solution sequence, when modified as indicated in the following section, provides a new computational tool that can be used to predict both the time at which the onset of dynamic buckling occurs and the dynamic buckling mode shapes of an impulsively loaded structure.

Dynamic Buckling Analysis

Linear buckling analysis requires solution of the eigenvalues problem:

$$\left[[K] + \lambda[K_\sigma]\ \{\phi\} = 0\right] \tag{1}$$

where

[K] structural stiffness matrix;

$[K_\sigma]$ stress stiffness matrix

λ, $\{\phi\}$ denote the associated eigenvalue and eigenvector

In terms of the buckling analysis, the eigenvector $\{\phi\}$ represents the buckling mode shape, and the associated eigenvalue λ indicates the multiple of $[K_\sigma]$ needed to make equation (1) singular, that is, to cause buckling. In a one-dimensional column buckling problem, each scalar eigenvalue satisfying equation (1) physically represents the nondimensional ratio:

$$\lambda = \frac{\sigma A}{P_*} \tag{2}$$

where σ is the compressive stress in the column, A is the cross-sectional area, and P_* is the buckling load. If the eigenvalue has the critical value of unity ($\sigma A = P_*$), buckling in the associated mode occurs.

In the dynamic case, the terms of $[K_\sigma]$ in Eq. (1) vary with time as the stress waves propagate through the structure. The eigensolution of (1) then becomes time dependent, and can be used to track the buckling stability as a function of time. Figure 1 is a simplified representation of a modified direct-time integration solution sequence in which the updated stress stiffness matrix is formed after each time step Δt, and the associated eigenvalue problem in equation (1) is solved. The eigenvalue is now a function of time, and it indicates the onset of buckling when it reaches the critical value of unity. Figure 2 is the DMAP alter which incorporates this dynamic buckling algorithm into the existing transient solution sequence.

DMAP Procedure

The functions of the DMAP statements shown in Fig. 2 are summarized here. In line 2 the number of columns in the UPV matrix is determined. This matrix contains the displacement, velocity and acceleration vectors for each degree of freedom at each time step. Lines 2 through 16 follow the Bubble Algorithm approach of Ref. 13. The DMI column matrices TIP1 and BAS1 from the Bulk Data deck, each initially sized to contain more rows than columns in the UPV matrix, are used to form two new column matrixes, MNTRJ and BOOTI. The number of rows in each of these matrices is equal to the number of columns in the UPV matrix. The monitor matrix MNTRJ initially contains unity in the first row and zero in the remaining rows. The BOOTI matrix always contains unity in the last row and zero in the remaining rows.

Having determined the size of the partitioning matrices, the eigenvalue extraction data is determined in line 19 and the buckling calculations are now performed. At the beginning of each pass through the RAALOOP, corresponding to each integration time step of the requested output, the current column position is compared with the number of columns in the UPV matrix, lines 25 through 27, ending the loop at the end of the available data. Continuing within the loop the unity value of the MNTRJ matrix is advanced three rows, lines 28 through 31, pointing to the location of the current displacement vector in the UPV matrix. The MNTRJ matrix is used to partition the UPV matrix, line 32, stripping the column containing the displacements. These displacements are used in the DSMG1 module, line 33, to form the time-varying global differential stiffness matrix, KDGG. The reduced differential stiffness matrix, KDAA, is then formed by eliminating the restrained and dependent degrees of freedom, line 35 through 45, and in line 47 this matrix is multiplied by negative one, forming the KDAAM matrix. The stiffness matrices KAA and KDAAM are then used in the READ module, line 48, to solve for the eigenvalues and eigenvectors for each integration time step initially requested for output.

The eigenvalue for each time step is printed by line 52. Optionally, lines 53 and 54 may be used to print eigenvalues and eigenvalue extraction data. Line 58 may be used to print eigenvectors. The RAALOOP is ended at line 64.

The computationally intensive nature of this analysis can be made more efficient by slightly modifying the DAMP procedure. A promising method is to perform the buckling analysis at specified time intervals in the transient solution sequence rather than after every time step, as is done here. The length of the time interval can be progressively decreased as the eigenvalue begins to change more rapidly, or as the critical value of unity is approached.

This technique will significantly reduce the number of individual buckling analyses performed, and hence will result in a more computationally efficient algorithm.

Example Problem

In order to establish the validity of this analysis procedure, a simple problem with a known solution, as given in Ref. 14, was analyzed. The propagation of a longitudinal compressive pulse in a long prismatic bar, shown in Fig. 3, was modelled.

Assuming a one-inch diameter aluminum bar of uniform circular cross section the elastic and geometric constants are:

$$E = 10 \times 10^6 \text{ psi} \tag{3}$$

$$I = \frac{\pi r^4}{4} = \frac{\pi}{64} \text{ in.}^4 \tag{4}$$

$$A = \pi r^2 = \frac{\pi}{4} \text{ in.}^2 \tag{5}$$

$$\rho = 2.5\times 10^{-4} \frac{\text{lb} \cdot \text{s}^2}{\text{in.}^4} \tag{6}$$

$$L = 100 \text{ in.} \tag{6}$$

where E is the Young's Modulus, I is the area moment of inertia, A is the cross-sectional area, ρ is the mass density, and L is the length of the bar.

The lowest buckling load is given by (Ref. 15):

$$P_* = \frac{\pi^2 EI}{4L^2} = 121 \text{ lb} \tag{7}$$

As shown in Fig. 3, the applied load is identical to the static buckling load in Eq. (7).

Using the above material constants, the bar wave velocity is given by (Ref. 14):

$$C_o = \sqrt{\frac{E}{\rho}} = 200{,}000 \frac{\text{in.}}{\text{sec}} \tag{8}$$

so the time for the longitudinal compression wave to travel from the impact point to the distal end of the bar is

$$t_o = \frac{L}{C_o} = 500 \ \mu\text{s} \tag{9}$$

A NASTRAN model consisting of ten rod elements, for a total of ten unconstrained axial degrees of freedom, was used to model the longitudinal impact of the bar. The integration time step was taken as

$$\Delta t = \frac{1}{4}\,\frac{L}{10C_o} = 12.5\ \mu s \tag{10}$$

to insure a numerically converged solution. The propagation of the compression wave from the point of impact to the clamped end of the bar is depicted in Figs. 4(a) and (b).

The compressive pulse, traveling at a speed C_o, reaches the complete length of the bar at time t_o (500 μs). Because the distal end of the bar is held fixed, the incident compressive pulse reflects (Ref. 15) as a pulse of the same sign (compressive) which superimposes on the existing uniform compressive stress in the bar. Figures 4(c) and (d) depict the progression of the reflected pulse, traveling at a speed C_o, back to the proximal end of bar, effectively doubling the compressive load supported by the bar. Reflecting from the proximal (free) end as a pulse of opposite sign (tensile) which superimposes on the existing compressive stress, the bar returns to its original fully stressed state at time $3t_o$, (1500 μs) as shown in Figs. 4(e) and (f). Finally, in Figs. 4(g) and (h), the tensile pulse reflects as a tensile pulse from the fixed end which temporarily cancels the uniform compression at time $4t_o$ (2000 μs), leaving the bar instantaneously unstressed. The stress states depicted in Figs. 4(i) and (j), for all practical purposes identical to those in Figs. 4(a) and (b), indicate that, assuming no damping exists, the above cycle will repeat itself indefinitely.

The corresponding time dependence of the lowest eigenvalue is shown in Fig. 5. The critical value of 1.0 is reached at times t_o, $3t_o$, $5t_o$, $7t_o$,. . . (500, 1500, 2500, 3500 μs,. . .); and whenever the bar supports a uniform compressive stress corresponding to its buckling load. Similarly, the eigenvalue reaches to its lower limit of 0.5 at times $2t_o$, $6t_o$, $10t_o$,. . . (1000, 3000, 5000 μs,. . .); and whenever the stress state is double that of the buckling load. The eigenvalue becomes large (theoretically infinite) at time 0, $4t_o$, $8t_o$,. . . (0, 2000, 4000, 6000 μs,. . .) ; and whenever the bar is unstressed.

Superimposed on the finite element results in Fig. 4 is the theoretical 1-D solution, assuming the stress wave propagates nondisperively at a constant speed C_o and reflects from the boundaries as described above. Good agreement exists between the two solutions, even when relatively few finite elements are used to model the bar. The time behavior of the lowest eigenvalue, shown in Fig. 5, can be interpreted directly in terms of the transient stress distribution in Fig. 4. Since the applied compressive load is exactly equal to the first static buckling load in Eq. (7), and no strain-rate dependence was assumed in the finite element model, buckling is predicted whenever the bar is uniformly stressed with its critical static buckling stress, which occurs at odd multiples of t_o, as shown in Fig. 4.

In a practical application, the above analysis is valid only until the onset of buckling occurs, since no post-buckling behavior has yet been included in the finite element model. The time integration was extended in the example problem only to physically interpret the results of the dynamic buckling analysis.

The example problem could have been solved without the use of a finite element analysis because of the simple non-dispersive nature of the longitudinal wave propagation. However, the propagation of flexural waves in beam-like structures is dispersive by nature, and as such would pose a formidable challenge without the use of some type of computational simulation. In Ref. 11, experimental measurements of delamination duckling in graphite/epoxy composite laminates were reported. The beam-like experimental specimens had simulated delaminations (ply disbonds) embedded in them during the fabrication process. They were held clamped at both ends and impacted transversely, as depicted schematically in Fig. 6. The subsequent flexure-induced local buckling of the delamination was recorded using strain gages and high speed photography. A finite element model of the initially flawed experimental specimen is used here to verify that the dynamic delamination buckling phenomenon can be predicted using computational simulation. Figure 6 shows the geometry and loading conditions for the initially flawed composite laminate subjected to a transverse impact. The finite element discretization of this laminate near the embedded flaw is shown schematically in Fig. 7. The layered structure of the composite laminate is represented by layers of shell elements. Multipoint constraints are imposed on the degrees of freedom between neighboring nodal points in the thickness direction such that simple beam bending displacements are enforced; that is, plane sections remain plane and no strain exists in the thickness direction. These constraints are removed in the delaminated region to allow the delaminated plies to separate from the main laminate when a local compression occurs in that area, as shown in Fig. 7. More complete details of the finite element modeling procedure are given in Ref. 12.

The progression of the flexural waves out from the central impact point to the boundaries of the laminate are shown in Fig. 8. As the disturbance passes through the flawed region at 100 to 150 μs after impact, the delaminated ligament separates from the laminate and begins to support a compressive longitudinal stress which increases in magnitude until it causes a local buckling of the delamination. The eigenvalue behavior and corresponding buckling mode are shown in Fig. 9. As the laminate deforms under the applied load, the eigenvalue decreases monotonically in magnitude until it reaches the critical value of unity, indicating the onset of buckling at approximately 190 μs from impact. The corresponding buckling mode shape is also depicted in the figure.

These results correspond closely with experimental observations. Both the buckling mode shape and the time at which buckling occurs are in good agreement with measurements taken from high speed photographs. A detailed comparison of finite element results and experimental measurements is given in Ref. 11.

CONCLUSIONS

A dynamic delamination buckling analysis procedure has been incorporated, in the form of a DMAP alter, into the transient analysis rigid format of NASTRAN. With this enhancement, NASTRAN can be used to calculate the time at which dynamic buckling occurs and the buckling mode shape of a structure subjected to dynamic loading. Comparison of the calculated results with a known solution supports the validity of the analysis. Application of the dynamic buckling analysis to the more complex problem of transverse impact of beam-like laminate was demonstrated, and the results phenomenologically duplicated those reported in earlier experiments.

REFERENCES

1. Gillespie, J.W.; and Pipes, R.B.: Compressive Strength of Composite Laminates with Interlaminar Defects. *Composite Structures*, vol. 2, no. 1, 1984, pp. 49-69.

2. Rosenfeld, M.S.; and Gause, L.W.: Compression Fatigue Behavior of Graphite/Epoxy in the Presence of Stress Raisers. *Fatigue of Fibrous Composite Materials*, ASTM STP-723, K.N. Lauraitis, ed., American Society for Testing and Materials, 1981, pp. 174-196.

3. Konishi, D.Y.; and Johnston, W.R.: Fatigue Effects on Delaminations and Strength Degradation in Graphite/Epoxy Laminates. *Composite Materials: Testing and Design (Fifth Conference)*, ASTM STP-674, S.W. Tsai, ed., American Society for Testing and Materials, 1979, pp. 597-619.

4. Chai, H.; Knauss, W.G.; and Babcock, C.D.: Observation of Damage Growth in Compressively Loaded Laminates. Exper. Mech., vol. 23, no. 3, Sept. 1983, pp. 329-337.

5. Simitses, G.J.; and Sallam, S.: Delamination Buckling and Growth of Flat Composite Structural Elements. AFOSR-85-1067TR, Sept. 1984. (Avail. NTIS, AD-A162370).

6. Kardomateas, G.A.; and Schmueser, D.W.: Buckling and Postbuckling of Delaminated Composites under Compressive Loads Including Transverse Shear Effects. AIAA J., vol. 26, no. 3, Mar. 1988, pp. 337-343.

7. Whitcomb, J.D.: Finite Element Analysis of Instability Related Delamination Growth. J. Compos. Mater., vol. 15, no. 5, Sept. 1981, pp. 403-426.

8. Kapania, R.K.; and Wolfe, D.R.: Delamination Buckling and Growth in Axially Loaded Beam-Plates. 28th Structures, Structural Dyanmics, and Materials Conference, Part 1, AIAA, 1987, pp. 766-775.

9. Whitcomb, J.D.; and Shivakumar, K.N.: Strain-Energy Release Rate Analysis of a Laminate with a Postbuckled Delamination. NASA TM-89091, 1987.

10. Sun, C.T.; and Grady, J.E.: Dyanmic Delamination Crack Propagation in a Graphite/Epoxy Laminate. Composite Materials: Fatigue and Fracture, ASTM STP-907, H.T. Hahn, ed., American Society for Testing and Materials, 1986, pp. 5-31.

11. Grady, J.E.; and De Paula, K.J.: Measurement of Impact-Induced Delamination Buckling in Composite Laminates. Dynamic Failure, Proceedings of the 1987 SEM Fall Conference, Society for Experimental Mechanics, Bethel, CT, 1987, pp. 160-168.

12. Grady, J.E.; Chamis, C.C.; and Aiello, R.A.: Dyanamic Delamination Buckling in Composite Laminates under Impact Loading: Computational Simulation. NASA TM-100192, 1987.

13. Pamidi, P.R.; and Butler, T.G.: Bubble Vector in Automatic Merging. Fifteenth NASTRAN Users' Colloquim, NASA CP-2481, 1987, pp. 118-135.

14. Graff, K.F.: Wave Motion in Elastic Solids. Ohio State University Press, 1975, pp. 75-139.

15. Timoshenko, S.P.: Strength of Materials. 2nd ed., D. Van Nostrand Company, Inc., 1941.

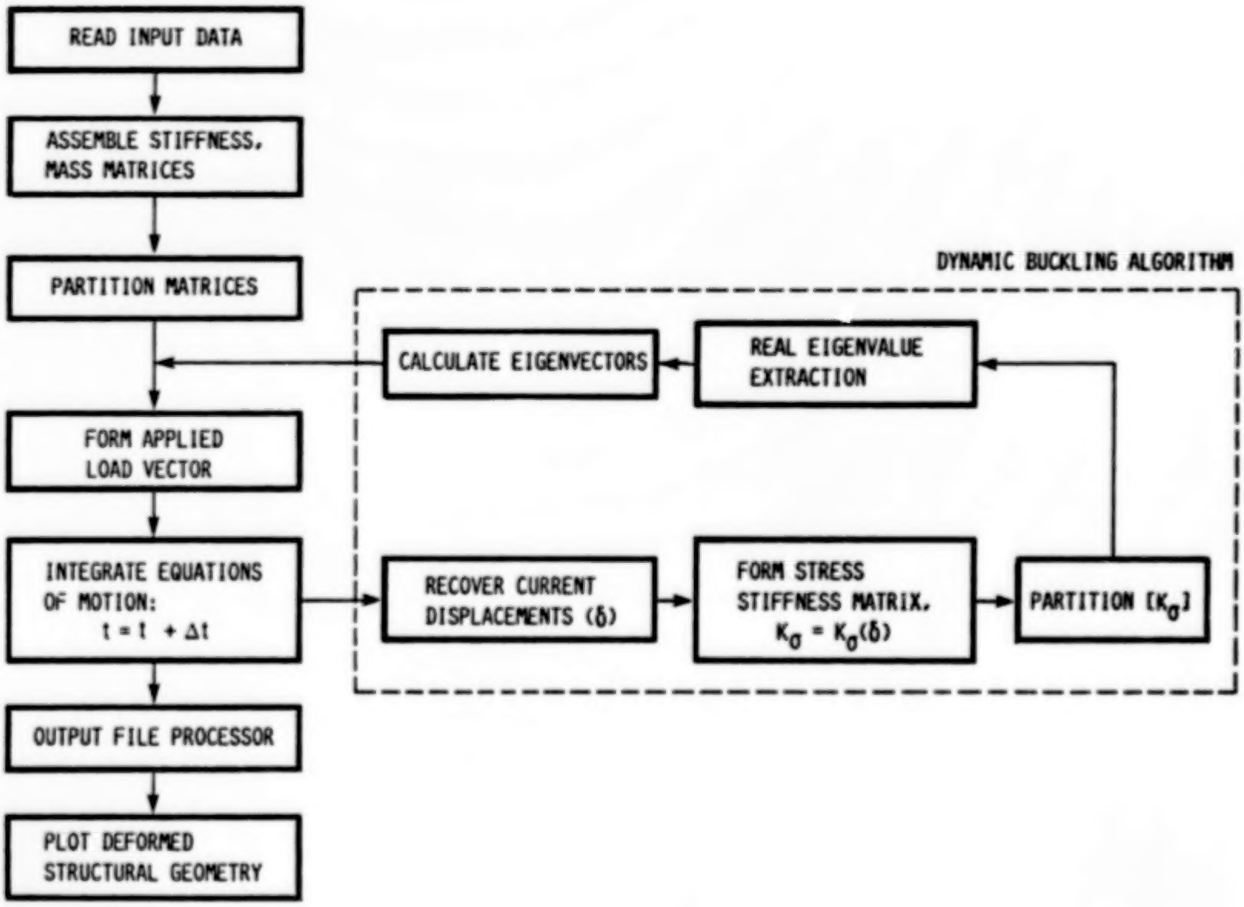
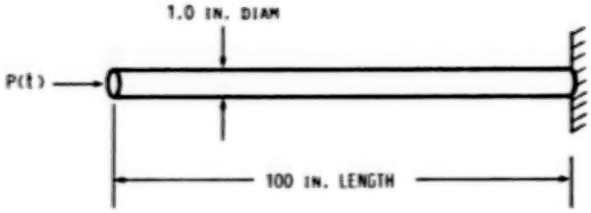

READ INPUT DATA
ASSEMBLE STIFFNESS, MASS MATRICES
PARTITION MATRICES
DYNAMIC BUCKLING ALGORITHM
CALCULATE EIGENVECTORS
REAL EIGENVALUE EXTRACTION
FORM APPLIED LOAD VECTOR
INTEGRATE EQUATIONS OF MOTION: t = t + Δt
RECOVER CURRENT DISPLACEMENTS (δ)
FORM STRESS STIFFNESS MATRIX, K_σ = K_σ(δ)
PARTITION [K_σ]
OUTPUT FILE PROCESSOR
PLOT DEFORMED STRUCTURAL GEOMETRY

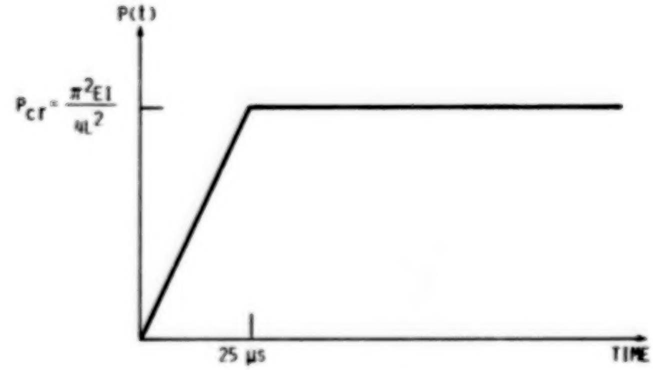
FIGURE 1. - DYNAMIC BUCKLING ANALYSIS SOLUTION SEQUENCE.

```
ALTER 146       $$$$$ COSMIC NASTRAN RF 09    RELEASE 1987    $$$$$
PARAML    UPV//*TRAILER*/1/V,N,NOCUPV $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
COPY      TIP1 / CLUSI/ 0 $                                                      $
COPY      TIP1 / BUBLI/ 0 $                                                      $
PARAM     //*SUB*/SHIFT/NOCUPV/ 1 $                                              $
LABEL     BUBTOP $<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<$  BUTLER/PAMIDI      $
FILE      BUBLI=SAVE/ CLUSI=SAVE $                         $                     $
PARTN     BUBLI,,BAS1/DUMMY,,,/7 $                         $     BUBBLE          $
MERGE     DUMMY,,,,TIP1/BUBLJ/7 $                          $                     $
ADD       CLUSI,BUBLJ/CLUSJ/ $                             $    ALGORITHM        $
SWITCH    BUBLJ,BUBLI//-1 $                                $                     $
SWITCH    CLUSJ,CLUSI//-1 $                                $                     $
REPT      BUBTOP,SHIFT $>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>$                     $
PARTN     TIP1,,CLUSJ/,MNTRI,,/7 $                                               $
PARTN     BUBLJ,,CLUSJ/,BOOTI,,/7 $                                              $
COPY      MNTRI/MNTRJ/ 0 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
TA1       ECT,EPT,BGPDT,SIL,GPTT,CSTM,/X1,X2,X3,ECPT,GPCT,,,/LUSET/
          NOSIMP/0/NOGENL/GENEL $
DPD       DYNAMICS,GPL,SIL,USET/GPLD,SILD,USETD,,,,,,,EED,EQDYN/V,N,
          LUSET/V,N,LUSETD/V,N,NOTFL/V,N,NODLT/V,N,NOPSDL/V,N,NOFRL/
          V,N,NONLFT/V,N,NOTRL/S,N,NOEED/C,N,123/V,N,NOUE $
COND      ERROR5,NOEED $
PARAM     //*NOP*/V,N,COLNUM=1 $
LABEL     RAALOOP $<<<<<<<<<<<<<<<<       TOP OF RAALOOP       <<<<<<<<<<<<<<<<<<$
PARAM     //*ADD*/COLNUM/COLNUM/3 $                                              $
PARAM     //*LE*/V,N,GETOUT/NOCUPV/COLNUM $                                      $
COND      QUITRAA,GETOUT $                                                       $
LABEL     CORTOP $<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<$    $
PARTN     MNTRJ,,BOOTI/DUM11,,,/7 $            MOVE THE COLUMN MARKER       $    $
MERGE     DUM11,,,,MNTRI/MNTRJ/7 $             THREE PLACES                 $    $
REPT      CORTOP, 2  $>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>$    $
PARTN     UPV,MNTRJ,/,,COLUPV,/1  $                                              $
DSMG1     CASECC,GPTT,SIL,EDT,COLUPV,CSTM,MPT,ECPT,GPCT,DIT/
          KDGG/DSCOSET $                                                         $
EQUIV     KDGG,KDNN/MPCF2 $                                                      $
COND      LBL2D,MPCF2 $                                                          $
MCE2      USET,GM,KDGG,,,/KDNN,,, $                                              $
LABEL     LBL2D $                                                                $
EQUIV     KDNN,KDFF/SINGLE $                                                     $
COND      LBL3D,SINGLE $                                                         $
SCE1      USET,KDNN,,,/KDFF,KDFS,,,, $                                           $
LABEL     LBL3D $                                                                $
EQUIV     KDFF,KDAA/OMIT $                                                       $
COND      LBL5D,OMIT $                                                           $
SMP2      USET,GO,KDFF/KDAA $                                                    $
LABEL     LBL5D $                                                                $
ADD       KDAA,/KDAAM/C,N,(-1.0,0.0)/C,N,(0.0,0.0) $                             $
READ      KAA,KDAAM,,,EED,USET,CASECC/LAMA,PHIA,MI,OEIGS/C,N,BUCKLING/
          S,N,NEIGV/C,N,2 $                                                      $
COND      ERROR4,NEIGV $                                                         $
PARAML    LAMA//*DTI*/2/3/V,N,EIGV $                                             $
PRTPARM   //0/*EIGV*   $                                                         $
$OFP       OEIGS,LAMA,,,,//S,N,CARDNO $                                          $
$OFP       LAMA,,,,,//S,N,CARDNO $                                               $
SDR1      USET,,PHIA,,,GO,GM,,KFS,,/PHIG,,BQG/C,N,1/C,N,BKL1 $                   $
SDR2      CASECC,CSTM,MPT,DIT,EQEXIN,SIL,,,BGPDT,LAMA,BQG,PHIG,EST,,/,
          OBQG1,OPHIG,OBES1,OBEF1,PPHIG/C,N,BKL1 $                               $
$OFP       OPHIG,OBQG1,OBEF1,OBES1,,//S,N,CARDNO $                               $
COND      P3,JUMPPLOT $                                                          $
PLOT   PLTPAR,GPSETS,ELSETS,CASECC,BGPDT,EQEXIN,SIL,,PPHIG ,GPECT,OBES1/
          PLOTX3/V,N,NSIL/V,N,LUSET/V,N,JUMPPLOT/V,N,PLTFLG/S,N,PFILE            $
PRTMSG    PLOTX3 // $                                                            $
LABEL     P3 $                                                                   $
REPT      RAALOOP,1000 $>>>>>>>>        BOTTOM OF RAALOOP      >>>>>>>>>>>>>>>>>>$
JUMP      QUITRAA $
LABEL     ERROR5 $
PRTPARM   //C,N,-3/C,N,BUCKLING $
JUMP      QUITRAA $
LABEL     ERROR4 $
PRTPARM   //C,N,-4/C,N,BUCKLING $
LABEL     QUITRAA $
JUMP      FINIS $
ENDALTER


BULK DATA
DMI       TIP1      0        2        1        1                  1000     1
DMI       TIP1      1        1        1.0
DMI       BAS1      0        2        1        1                  1000     1
DMI       BAS1      1        1000     1.0
```

FIGURE 2. - DYNAMIC BUCKLING DMAP ALTER FOR TRANSIENT SOLUTION SEQUENCE.


1.0 IN. DIAM
P(t)
100 IN. LENGTH

(A) GEOMETRY.



(B) LOADING.

FIGURE 3. - EXAMPLE PROBLEM GEOMETRY AND LOADING: LONGITUDINAL IMPACT OF A UNIFORM BAR.

BEST COPY AVAILABLE

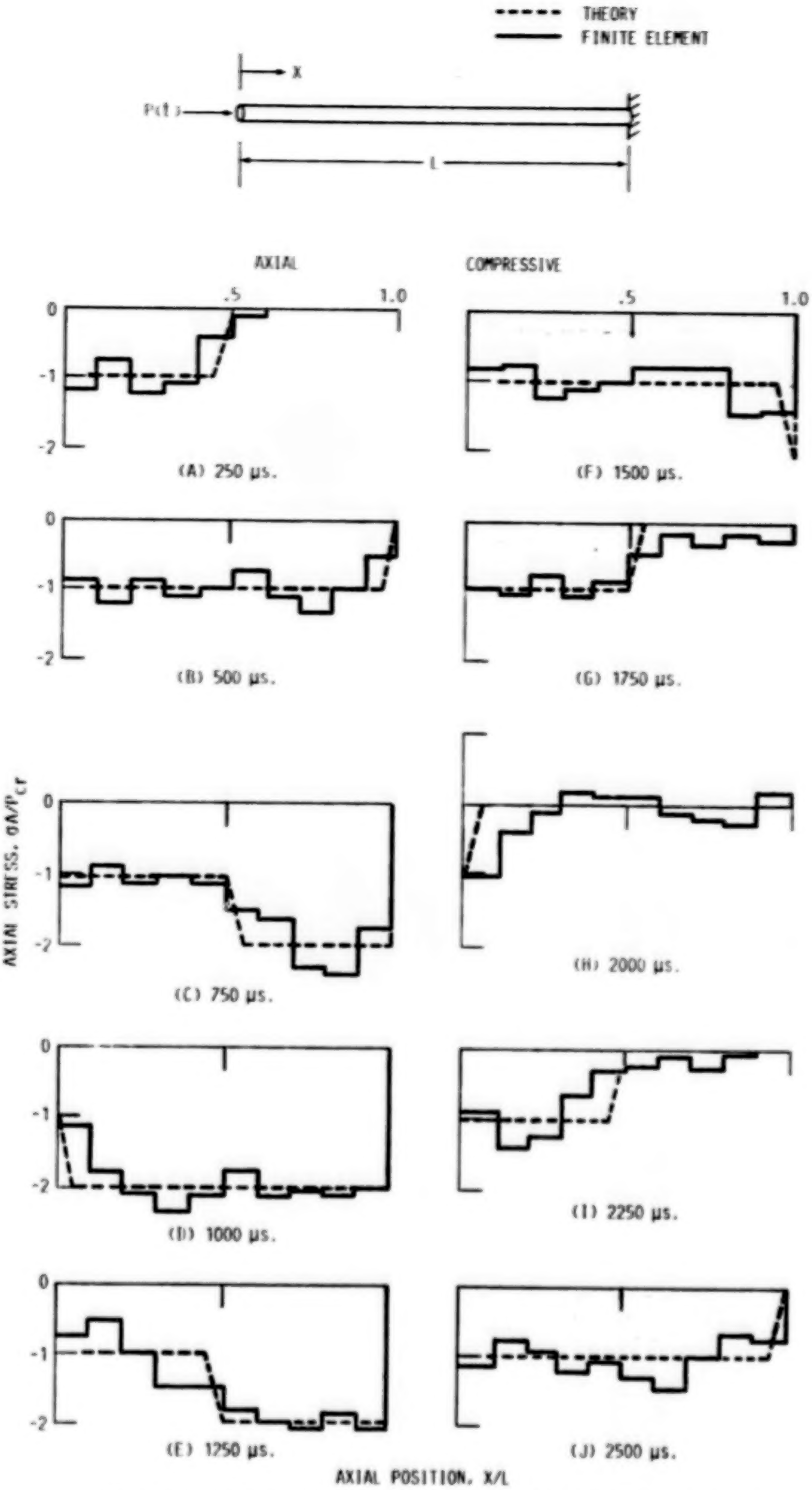


FIGURE 4. - AXIAL STRESS DISTRIBUTION IN BAR DUE TO LONGITUDINAL IMPACT.

BEST COPY
AVAILABLE

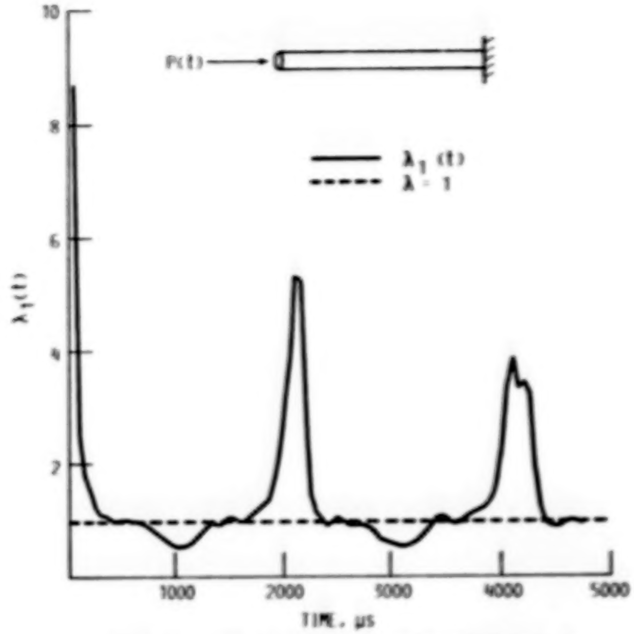


FIGURE 5. - TIME DEPENDENCE OF FIRST EIGENVALUE IN EXAMPLE PROBLEM.

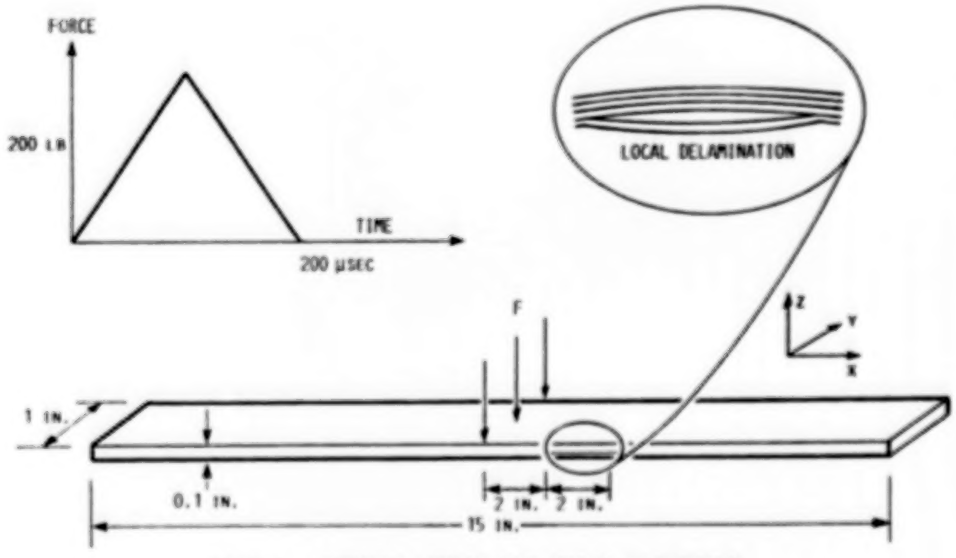

FORCE
200 LB
TIME
200 µSEC
LOCAL DELAMINATION
F
Z
Y
X
1 IN.
0.1 IN.
2 IN.
2 IN.
15 IN.

FIGURE 6. - COMPOSITE LAMINATE WITH INITIAL DELAMINATION.

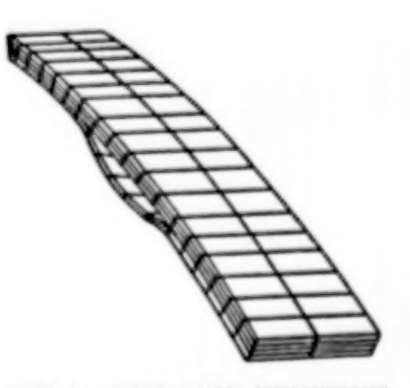


FIGURE 7. - FINITE ELEMENT REPRESENTATION OF COMPOSITE LAMINATE SHOWING LOCAL BUCKLING OF THE DELAMINATED PLIES.

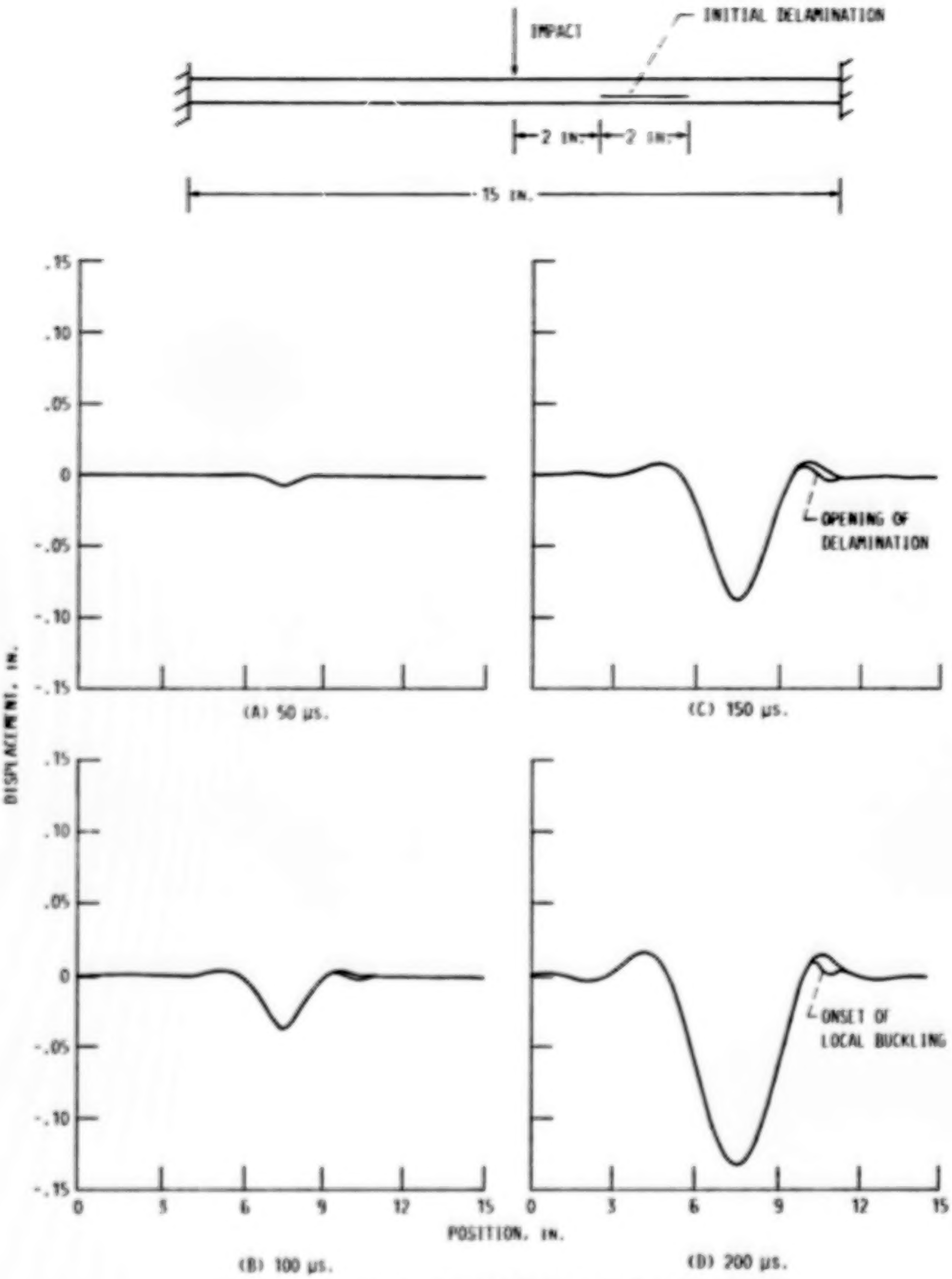


FIGURE 8. - MIDPLANE DISPLACEMENT PROFILES OF IMPACTED LAMINATE.

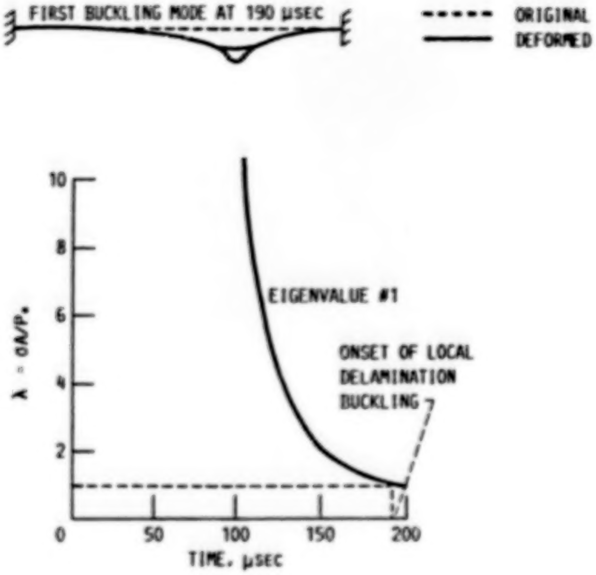


FIGURE 9. - TIME DEPENDENCE OF LOWEST EIGENVALUE FOR COMPOSITE LAMINATE SUBJECTED TO TRANSVERSE IMPACT.

IMPROVEMENTS TO THE CONTINUE FEATURE IN TRANSIENT ANALYSIS

by

P. R. Pamidi
RPK Corporation
Columbia, Maryland

SUMMARY

The CONTINUE feature in transient analysis as implemented in the standard release of COSMIC/NASTRAN has inherent errors associated with it. As a consequence, the results obtained by a CONTINUEd restart run do not, in general, match the results that would have been obtained in a single run without the CONTINUE feature. These inherent errors have been eliminated by improvements to the restart logic that have been developed by RPK Corporation and that are available on all RPK-supported versions of COSMIC/NASTRAN. These improvements ensure that the results of a CONTINUEd transient analysis run are the same as those of a non-CONTINUEd run. In addition, the CONTINUE feature has been extended to transient analysis involving uncoupled modal equations. The improvements and enhancement have been illustrated by examples.

INTRODUCTION

In transient analysis, it is frequently necessary to continue the integration of equations beyond the last (or from any earlier intermediate) output time for which the results were obtained in a previous run. Thus, the initial time for the new run is to be a specified output time of the previous run and the initial conditions for the new run are to be the same as the conditions existing at the specified output time of the previous run. The CONTINUE feature in NASTRAN makes it possible to do this without re-executing the entire problem.

The CONTINUE feature involves performing a checkpoint run of a transient analysis problem followed by a restart run with the aim of continuing the integration from the last (or from any earlier intermediate) output time for which the results were obtained in the checkpoint run. The details of the usage of this feature are given in Vol. 2, Section 2.9.6 of the NASTRAN User's Manual (Reference 1). The theoretical aspects of this feature are discussed in Section 11.4.2 of the NASTRAN Theoretical Manual (Reference 2) and the programming aspects are detailed in Section 4.65.7.3 of the NASTRAN Programmer's Manual (Reference 3).

EQUATIONS FOR TRANSIENT ANALYSIS

The equations used in transient analysis are discussed in detail in References 2 and 3. For the sake of convenience, they are summarized in the Appendix.

The most important equation employed in transient analysis is Equation (1) given in the appendix. After the integration is started, this equation is used to continue the integration by computing the displacements at successive times until there is a change in the time step. In order to compute the displacements at each time, this equation requires, in addition to the loads at that time, the displacements and the loads at the two previous times.

When there is a change in the time step employed for the integration, Equation (2) in the appendix is used to compute the displacements for the first time after the time step change. It is important to note that this equation is employed only once for each time step change. Equation (1) is then employed to resume the integration until there is another time step change. Every time Equation (2) is employed, it requires, in addition to the loads at that time, the displacements at the three previous times and the loads at the previous time.

CURRENT IMPLEMENTATION OF THE CONTINUE FEATURE

When the CONTINUE feature is employed in the standard release of COSMIC/NASTRAN to restart a transient analysis problem, Equation (3) in the appendix is used to start the integration from the specified time. Equation (1) or (2), as appropriate, is then used to continue the process. In order to start the integration, Equation (3) uses the displacements, velocities and accelerations at the specified output time of the checkpoint run. Unlike Equations (1) and (2), Equation (3) uses neither the loads at the specified output time nor any displacements at solution times prior to that output time.

The CONTINUE feature, as described above, is based on the inherent assumption that the output times of the checkpoint run do not include all of the solution times of the integration. In other words, it is assumed that the TSTEP bulk data card (Reference 1) in the checkpoint run involves one or more non-unity skip factors for output. As a result, the CONTINUE feature initiates the integration in the restart run by considering the conditions (namely, the displacements, velocities and accelerations) only at the specified output time of the previous run; it does not take into account the conditions at solution times just before that output time because the latter conditions are assumed to be not available to the restart run.

The above approach to the CONTINUE feature introduces inherent errors into the procedure (Reference 2). As a consequence, the results of a CONTINUEd restart run do not, in general, match the results that would have been obtained in a single run without using the CONTINUE feature.

It can also be seen from Equation (3) and the associated Equations (3a) through (3d) that the initial time step used in a CONTINUEd run can affect its results. As a result, the errors mentioned above may be magnified further if the initial time step in the CONTINUEd restart run is not the same as that used in the checkpoint run just before the restart (Reference 2).

In addition to the above deficiency, the current implementation of the CONTINUE feature has the further limitation that it is restricted to coupled equations and is not applicable to transient analysis involving uncoupled modal equations.

IMPROVED IMPLEMENTATION OF THE CONTINUE FEATURE

RPK has developed improvements to the CONTINUE feature that have removed the deficiencies described above. This involved extensive changes to the code in the TRD (Transient Analysis - Displacement approach) module, including, in particular, the addition of an extra output data block to this module, and minor related changes elsewhere. These changes also required minor modifications to the DMAP of Displacement Approach Rigid Formats 9 (Direct Transient Response) and 12 (Modal Transient Response). These improvements are available on all RPK-supported versions of COSMIC/NASTRAN, beginning with the 1988 release.

The highlights of RPK's improvements are described below.

1. The loads and displacements for all solution times of the checkpoint run, as well as the time steps employed to obtain each of those solutions, are made available to the restart run. With this information, the restart run using the CONTINUE feature now initiates the continuation of the integration using the same procedure as that used in a non-CONTINUEd run. This applies even in those cases where the time step used in the CONTINUEd restart run is different from that used in the earlier checkpoint run just before restart. As a result, Equation (3) is no longer used and the restart run uses Equation (1) or (2), as appropriate, to initiate the integration.

2. The key to providing the information required for the above procedure is the new data block output from the TRD module. There are two possible cases that need to be considered with regard to the generation of this data block.

   Case 1. The TSTEP bulk data card in the checkpoint run has no non-unity skip factors

   In this case, the output times, by definition, include all solution times. As a result, the displacements for all solution times are available from the standard UDVT (displacement-velocity-acceleration) matrix data block resulting from the TRD module in the checkpoint run. Accordingly, the new data block in this case is designed to contain only the loads for all solution times as well as the time steps used to obtain each of the solutions. This case is identified in the new data block (for subsequent use in the restart run) by setting the fourth word of its six-word trailer to 1.

   Case 2. The TSTEP bulk data card in the checkpoint run consists of one or more non-unity skip factors

   Since the output times in this case are only a subset of the solution times, the new data block is designed to contain not only the loads but also the displacements for all solution times, as well as the time steps used to obtain each of those solutions. Further, the new data block in this case has an additional special record that has information relating each output time to its corresponding solution time. This case is identified in the new data block by setting the fourth word of its six-word trailer to 0.

3. Using the information provided by the new data block described above, the restart run using the improved CONTINUE feature determines whether all of the loads and displacements

required for continuing the integration are available just from this data block alone (Case 2 above) or from both this data block and the UDVT data block (Case 1 above).

It should be emphasized that the new data block is generated only in checkpoint runs since its sole use is in a subsequent restart run using the CONTINUE feature. Also, it should be noted that, even when it is generated, this data block does not require any new computations since all of the information it needs is already available.

4. The restart run then determines if the time step to be used for continuing the integration in the restart run is the same as, or different from, the time step that was used for obtaining the solution for the specified output time of the checkpoint run. If the former condition is true, Equation (1) in the appendix is used to initiate the integration. If the latter condition is true, Equation (2) in the appendix is used to initiate the integration. Depending upon which of these two equations is used, the required loads and displacements for starting the integration are retrieved from the appropriate data block or data blocks mentioned earlier.

5. The procedure outlined above for the CONTINUE feature ensures that the CONTINUEd run starts the integration in the same manner as the integration would have continued had it proceeded further in the original checkpoint run. This therefore ensures that the results obtained by a CONTINUEd run will be the same as those of a non-CONTINUEd run.

6. Separately, RPK has extended the CONTINUE feature to transient analysis involving uncoupled modal equations. Since the equations in this case have closed-form solutions (References 2 and 3), this development introduces no errors.

EXAMPLES

In order to illustrate the improvements to the CONTINUE feature mentioned above, NASTRAN Demonstration Problem No. D09-01-1A (Transient Analysis with Direct Matrix Input) was selected. Two variations of this problem with the same standard input data, but with different sets of TSTEP bulk data input, were analyzed. These two cases are identified as Examples 1 and 2.

For both of the above examples, checkpoint and restart runs were made on the DEC VAX version using the standard release of COSMIC/NASTRAN and on RPK's CRAY version of COSMIC/NASTRAN using the improvements to the CONTINUE feature described in the paper. The results of these analyses are presented in Table 1 (for Example 1) and Table 2 (for Example 2). To facilitate comparison of the results, the restart runs were made from an intermediate output time of the checkpoint runs. It is quite clear from these tables that the results of the CONTINUEd restart runs on RPK's CRAY version match those of the non-CONTINUEd checkpoint runs perfectly, thereby validating the improvements. The results of the DEC VAX version do not exhibit the same correlation.

In order to illustrate the extension of the CONTINUE feature to transient analysis involving uncoupled modal equations, NASTRAN Demonstration Problem No. D12-01-1A (Transient Analysis of a Free One Hundred Cell Beam) was selected. This case is identified as Example 3.

Checkpoint and restart runs for the above problem were made on RPK's CRAY version mentioned above. (No runs were made on the standard release of COSMIC/NASTRAN as it does not support the CONTINUE feature for uncoupled modal equations.) Again, to facilitate comparison of the results, the restart run was made from an intermediate output time of the checkpoint run. The results are presented in Table 3. It is again quite clear from this table that the results of the CONTINUEd restart run on RPK's CRAY version match those of the non-CONTINUEd checkpoint run perfectly, thereby validating the new development.

CONCLUDING REMARKS

The CONTINUE feature in transient analysis as implemented in the standard release of COSMIC/NASTRAN has inherent errors associated with it. As a consequence, the results obtained by a CONTINUEd run do not, in general, match the results that would have been obtained in a single run without the CONTINUE feature. These inherent errors have been eliminated by improvements to the restart logic that have been developed by RPK Corporation and that are available on all RPK-supported versions of COSMIC/NASTRAN. These improvements ensure that the results of a CONTINUEd transient analysis run are the same as those of a non-CONTINUEd run. In addition, the CONTINUE feature has been extended to transient analysis involving uncoupled modal equations. The improvements and enhancement have been illustrated by examples.

REFERENCES

1. The NASTRAN User's Manual, NASA SP-222(08), June 1986.

2. The NASTRAN Theoretical Manual, NASA SP-221(06), January 1981.

3. The NASTRAN Programmer's Manual, NASA SP-223(05), December 1978.

TABLE 1. RESULTS FOR EXAMPLE 1

(Transient Analysis Involving Coupled Equations)

Displacements for Extra Point 10 of NASTRAN Demonstration Problem No. D09-01-1A

TSTEP Bulk Data Card Input

Checkpoint Run: Number of time steps = 200
Time increment = 0.010 sec.
Skip factor for output = 10

Number of time steps = 50
Time increment = 0.015 sec.
Skip factor for output = 5

Restart Run: Number of time steps = 100
Time increment = 0.010 sec.
Skip factor for output = 5

NOTES:

1. The restart run on the DEC VAX version uses the CONTINUE feature available in the standard release of COSMIC/NASTRAN.

2. The restart run on RPK's CRAY version uses the improved CONTINUE feature described in the paper. In this case, since the initial time step used in the restart run (0.010 sec.) is the same as that used to obtain the output of the checkpoint run from where the restart is initiated, the CONTINUE features uses Equation (1) in the appendix to initiate the integration.

3. The restart run was initiated by setting the parameter NCOL in the rigid format DMAP to 11 just before the TRLG module (see Volume 2, Section 2.9.6 of Reference 1 for details), thereby triggering the continuation of the integration from the 11th output time of the checkpoint run.

4. Note that the time step changes from 0.010 sec. to 0.015 sec. at time = 2.000 sec.

5. The % Error in the table is calculated by the following relationship:

$$\frac{\text{(Displacement from Restart Run) - (Displacement from Checkpoint Run)}}{\text{(Displacement from Checkpoint Run)}} \text{ X } 100$$

TABLE 1. RESULTS FOR EXAMPLE 1
(Continued)

	Results from DEC VAX Version of 1988 COSMIC/NASTRAN (See Note 1 above)			Results from RPK's CRAY Version of 1988 COSMIC/NASTRAN (See Note 2 above)		
Time (sec.) (See Note 4 above)	Displacements from Checkpoint Run	Displacements from Restart Run	% Error (See Note 5 above)	Displacements from Checkpoint Run	Displacements from Restart Run	% Error (See Note 5 above)
.000	0.000000E+00	See Note 3 above		0.000000E+00	See Note 3 above	
.100	8.404462E-01			8.404462E-01		
.200	9.099538E-01			9.099538E-01		
.300	1.447636E-01			1.447636E-01		
.400	-7.532177E-01			-7.532178E-01		
.500	-9.602748E-01			-9.602749E-01		
.600	-2.864749E-01			-2.864749E-01		
.700	6.501075E-01			6.501076E-01		
.800	9.903484E-01			9.903484E-01		
.900	4.221459E-01			4.221458E-01		
1.000	-5.332897E-01	-5.332897E-01	.00	-5.332899E-01	-5.332899E-01	.00
1.100	-9.995404E-01	-9.972498E-01	-.23	-9.995404E-01	-9.995404E-01	.00
1.200	-5.489158E-01	-5.464416E-01	-.45	-5.489157E-01	-5.489157E-01	.00
1.300	4.052275E-01	4.056157E-01	.10	4.052276E-01	4.052276E-01	.00
1.400	9.876569E-01	9.856030E-01	-.21	9.876569E-01	9.876569E-01	.00
1.500	6.641117E-01	6.614999E-01	-.39	6.641117E-01	6.641117E-01	.00
1.600	-2.686209E-01	-2.693950E-01	.29	-2.686212E-01	-2.686212E-01	.00
1.700	-9.549485E-01	-9.531747E-01	-.19	-9.549486E-01	-9.549486E-01	.00
1.800	-7.653049E-01	-7.626103E-01	-.35	-7.653047E-01	-7.653047E-01	.00
1.900	1.263505E-01	1.274941E-01	.91	1.263508E-01	1.263508E-01	.00
2.000	9.021050E-01	9.006485E-01	-.16	9.021051E-01	9.021051E-01	.00
2.015	9.624534E-01	9.606622E-01	-.19	9.624535E-01	9.624535E-01	.00
2.075	9.827350E-01	9.800652E-01	-.27	9.827349E-01	9.827349E-01	.00
2.150	5.388174E-01	5.363594E-01	-.46	5.388172E-01	5.388172E-01	.00
2.225	-1.927009E-01	-1.936352E-01	.48	-1.927012E-01	-1.927012E-01	.00
2.300	-8.213626E-01	-8.202745E-01	-.13	-8.213628E-01	-8.213628E-01	.00
2.375	-1.011611E+00	-1.009082E+00	-.25	-1.011611E+00	-1.011611E+00	.00
2.450	-6.618994E-01	-6.592784E-01	-.40	-6.618993E-01	-6.618993E-01	.00
2.525	4.110985E-02	4.242327E-02	3.19	4.111014E-02	4.111014E-02	.00
2.600	7.221763E-01	7.214810E-01	-.10	7.221765E-01	7.221765E-01	.00
2.675	1.017772E+00	1.015439E+00	-.23	1.017772E+01	1.017772E+00	.00
2.750	7.701184E-01	7.673931E-01	-.35	7.701182E-01	7.701182E-01	.00

TABLE 2. RESULTS FOR EXAMPLE 2

(Transient Analysis Involving Coupled Equations)

Displacements for Extra Point 10 of NASTRAN Demonstration Problem No. D09-01-1A

TSTEP Bulk Data Card Input

Checkpoint Run: Number of time steps = 200
Time increment = 0.005 sec.
Skip factor for output = 10

Number of time steps = 50
Time increment = 0.015 sec.
Skip factor for output = 5

Restart Run: Number of time steps = 50
Time increment = 0.015 sec.
Skip factor for output = 5

NOTES:

1. The restart run on the DEC VAX version uses the CONTINUE feature available in the standard release of COSMIC/NASTRAN.
2. The restart run on RPK's CRAY version uses the improved CONTINUE feature described in the paper. In this case, since the initial time step used in the restart run (0.015 sec.) is not the same as that used to obtain the output of the checkpoint run from where the restart is initiated (0.005 sec.), the CONTINUE feature uses Equation (2) in the appendix to initiate the integration.
3. The restart run was initiated by setting the parameter NCOL in the rigid format DMAP to 21 just before the TRLG module (see Volume 2, Section 2.9.6 of Reference 1 for details), thereby triggering the continuation of the integration from the 21st output time of the checkpoint run.
4. Note that the time step changes from 0.005 sec. to 0.015 sec. at time = 1.000 sec.
5. The displacement shown in the parentheses below is obtained only in the checkpoint run because this run involves a change in time step. There is no corresponding displacement from the restart run as the latter run does not involve any change in time step. In general, each change in time step results in such an extra displacement in the output.
6. The % Error in the table is calculated by the following relationship:

$$\frac{\text{(Displacement from Restart Run) - (Displacement from Checkpoint Run)}}{\text{(Displacement from Checkpoint Run)}} \text{ X } 100$$

TABLE 2. RESULTS FOR EXAMPLE 2
(Continued)

	Results from DEC VAX Version of 1988 COSMIC/NASTRAN (See Note 1 above)			Results from RPK's CRAY Version of 1988 COSMIC/NASTRAN (See Note 2 above)		
Time (sec.) (See Note 4 above)	Displacements from Checkpoint Run	Displacements from Restart Run	% Error (See Note 6 above)	Displacements from Checkpoint Run	Displacements from Restart Run	% Error (See Note 6 above)
.000	0.000000E+00	See Note 3 above		0.000000E+00	See Note 3 above	
.050	4.792385E-01			4.792386E-01		
.100	8.412145E-01			8.412146E-01		
.150	9.973578E-01			9.973578E-01		
.200	9.094625E-01			9.094625E-01		
.250	5.990351E-01			5.990351E-01		
.300	1.420328E-01			1.420328E-01		
.350	-3.497228E-01			-3.497229E-01		
.400	-7.559065E-01			-7.559065E-01		
.450	-9.771311E-01			-9.771311E-01		
.500	-9.592662E-01			-9.592662E-01		
.550	-7.066831E-01			-7.066831E-01		
.600	-2.811852E-01			-2.811851E-01		
.650	2.131146E-01			2.131147E-01		
.700	6.552684E-01			6.552685E-01		
.750	9.370877E-01			9.370878E-01		
.800	9.896156E-01			9.896156E-01		
.850	7.999993E-01			7.999992E-01		
.900	4.146350E-01			4.146349E-01		
.950	-7.218431E-02			-7.218443E-02		
1.000	-5.413412E-01	-5.413412E-01	.00	-5.413413E-01	-5.413413E-01	.00
1.015	(-6.622212E-01)	See Note 5		(-6.622213E-01)	See Note 5	
1.075	-9.761183E-01	-9.721630E-01	-.41	-9.761184E-01	-9.761184E-01	.00
1.150	-8.898796E-01	-8.838871E-01	-.67	-8.898796E-01	-8.898796E-01	.00
1.225	-3.286562E-01	-3.238251E-01	-1.47	-3.286560E-01	-3.286560E-01	.00
1.300	4.079918E-01	4.090828E-01	.27	4.079919E-01	4.079919E-01	.00
1.375	9.268688E-01	9.236374E-01	-.35	9.268689E-01	9.268689E-01	.00
1.450	9.510176E-01	9.451886E-01	-.61	9.510176E-01	9.510176E-01	.00
1.525	4.675484E-01	4.622332E-01	-1.14	4.675483E-01	4.675483E-01	.00
1.600	-2.654808E-01	-2.674452E-01	.74	-2.654810E-01	-2.654810E-01	.00
1.675	-8.568062E-01	-8.543713E-01	-.28	-8.568063E-01	-8.568063E-01	.00
1.750	-9.908002E-01	-9.852656E-01	-.56	-9.908002E-01	-9.908002E-01	.00

TABLE 3. RESULTS FOR EXAMPLE 3

(Transient Analysis Involving Uncoupled Modal Equations)

Displacements for Grid Point 26, Component T3, of
NASTRAN Demonstration Problem No. D12-01-1A

TSTEP Bulk Data Card Input

Checkpoint Run: Number of time steps = 30
Time increment = 0.001388 sec.
Skip factor for output = 1

Restart Run: Number of time steps = 14
Time increment = 0.001388 sec.
Skip factor for output = 1

NOTES:

1. For the restart run of this problem, RPK's CRAY version uses the CONTINUE feature that has been extended to uncoupled modal equations and that is mentioned in the paper. No runs were made on the standard release of COSMIC/NASTRAN as it does not have this capability.

2. The restart run was initiated by setting the parameter NCOL in the rigid format DMAP to 17 just before the TRLG module (see Volume 2, Section 2.9.6 of Reference 1 for details), thereby triggering the continuation of the integration from the 17th output time of the checkpoint run.

3. The % Error in the table is calculated by the following relationship:

$$\frac{\text{(Displacement from Restart Run) - (Displacement from Checkpoint Run)}}{\text{(Displacement from Checkpoint Run)}} \times 100$$

TABLE 3. RESULTS FOR EXAMPLE 3
(Continued)

	Results from CRAY Version of 1988 COSMIC/NASTRAN (See Note 1 above)		
Time (sec.)	Displacements from Checkpoint Run	Displacements from Restart Run	% Error (See Note 3 above)
0.000000E+00	0.000000E+00	See Note 2 above	
1.388000E-03	3.346152E-03		
2.776000E-03	1.799760E-02		
4.164000E-03	3.245619E-02		
5.552000E-03	2.809689E-02		
6.940000E-03	3.562258E-03		
8.328000E-03	-3.010927E-02		
9.716000E-03	-6.037387E-02		
1.110400E-02	-7.236326E-02		
1.249200E-02	-5.503825E-02		
1.388000E-02	-1.209147E-02		
1.526800E-02	3.983471E-02		
1.665600E-02	8.073034E-02		
1.804400E-02	9.407332E-02		
1.943200E-02	7.293699E-02		
2.082000E-02	2.460154E-02		
2.220800E-02	-3.235601E-02	-3.235601E-02	.00
2.359600E-02	-7.679893E-02	-7.679893E-02	.00
2.498400E-02	-9.321987E-02	-9.321987E-02	.00
2.637200E-02	-7.703266E-02	-7.703266E-02	.00
2.776000E-02	-3.604619E-02	-3.604619E-02	.00
2.914800E-02	1.333591E-02	1.333591E-02	.00
3.053600E-02	5.347509E-02	5.347509E-02	.00
3.192400E-02	7.224338E-02	7.224338E-02	.00
3.331200E-02	6.658045E-02	6.658045E-02	.00
3.470000E-02	4.226934E-02	4.226934E-02	.00
3.608800E-02	1.031458E-02	1.031458E-02	.00
3.747600E-02	-1.823067E-02	-1.823067E-02	.00
3.886400E-02	-3.631594E-02	-3.631594E-02	.00
4.025200E-02	-4.226599E-02	-4.226599E-02	.00
4.164000E-02	-3.859226E-02	-3.859226E-02	.00

APPENDIX

The equations in this appendix are all taken from Section 4.65.7.3 of the NASTRAN Programmer's Manual (Reference 3). Readers are referred to that section for the complete definition and description of the various terms involved in the equations.

The following general terminology is used in all of the equations in this appendix:

Δt --- time step

u_i --- displacement at time t_i

$\dot{u}_i$ --- velocity at time t_i

$\ddot{u}_i$ --- acceleration at time t_i

P_i --- load at time t_i

N_i --- non-linear load at time t_i

[K] --- stiffness matrix

[B] --- damping matrix

[M] --- mass matrix

The matrices [C], [D] and [E] used in the equations in this appendix are functions of the [K], [B] and [M] matrices and the time step Δt and are defined as follows:

$$[C] = (2/\Delta t^2)\,[M] - (1/3)\,[K] \tag{a}$$

$$[D] = (1/\Delta t^2)\,[M] + (1/2\Delta t)\,[B] + (1/3)\,[K] \tag{b}$$

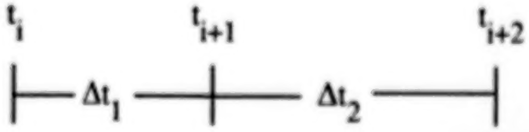
$$[E] = -(1/\Delta t^2)[M] + (1/2\Delta t)\,[B] - (1/3)\,[K] \tag{c}$$

Equation Used to Compute Successive Displacements After the Integration is Started

$$[D]\,\{u_{i+2}\} = (1/3)\,\{P_i + P_{i+1} + P_{i+2}\} + \{N_{i+1}\} + [C]\,\{u_{i+1}\} + [E]\,\{u_i\} \tag{1}$$

Equation Used to Compute the First Displacement After a Time Step Change

It is assumed here that the time step changes from Δt_1 to Δt_2 at time t_{i+1}. This is shown below.



The [C], [D] and [E] matrices are formed with using $\Delta t = \Delta t_2$.

$$[D]\{u_{i+2}\} = (1/3)\{P_i^1 + P_{i+1} + P_{i+2}\} + \{N_{i+1}\} + [C]\{u_{i+1}\} + [E]\{u_i^1\} \tag{2}$$

where:

$$\{\dot{u}_{i+1}\} = (1/\Delta t_1)(\{u_{i+1}\} - \{u_i\}) \tag{2a}$$

$$\{\ddot{u}_{i+1}\} = (1/\Delta t_1^2)(\{u_{i+1}\} - 2\{u_i\} + \{u_{i-1}\}) \tag{2b}$$

$$\{\dot{u}_i^1\} = \{\dot{u}_{i+1}\} - \{\ddot{u}_{i+1}\}\Delta t_2 \tag{2c}$$

$$\{u_i^1\} = \{u_{i+1}\} - \Delta t_2\{\dot{u}_{i+1}\} + (\Delta t_2^2/2)\{\ddot{u}_{i+1}\} \tag{2d}$$

and

$$\{P_i^1\} = [M]\{\ddot{u}_{i+1}\} + [B]\{\dot{u}_i^1\} + [K]\{u_i^1\} \tag{2e}$$

Equation Used by the CONTINUE Feature to Compute the First Displacement of the CONTINUEd Run

$$[D]\{u_1\} = (1/3)\{P_{-1} + P_0 + P_1\} + \{N_0\} + [C]\{u_0\} + [E]\{u_{-1}\} \tag{3}$$

where

$$\{P_0\} = [K]\{u_0\} + [B]\{\dot{u}_0\} + [M]\{\ddot{u}_0\} \tag{3a}$$

$$\{u_{-1}\} = \{u_0\} - \Delta t\{\dot{u}_0\} + (\Delta t^2/2)\{\ddot{u}_0\} \tag{3b}$$

$$\{\dot{u}_{-1}\} = \{\dot{u}_0\} - \{\ddot{u}_0\}\Delta t \tag{3c}$$

and

$$\{P_{-1}\} = [M]\{\ddot{u}_0\} + [B]\{\dot{u}_{-1}\} + [K]\{u_{-1}\} \tag{3d}$$

where $\{u_0\}$, $\{\dot{u}_0\}$ and $\{\ddot{u}_0\}$ are the displacements, velocities and accelerations, respectively, at the specified output time t_0 of the checkpoint run and Δt is the initial time step for the restart. $\{P_1\}$ is the load at time $t_0 = t_0 + \Delta t$ and $\{N_0\}$ is the initial non-linear load.

FINITE ELEMENT MODELING OF ELECTROMAGNETIC FIELDS AND WAVES USING NASTRAN

E. Thomas Moyer Jr.
George Washington University
Washington, D.C 20052

Erwin Schroeder
David Taylor Research Center
Bethesda, Md. 20084

SUMMARY

The various formulations of Maxwell's equations are reviewed with emphasis on those formulations which most readily form analogies with Navier's equations. Analogies involving scalar and vector potentials and electric and magnetic field components are presented. Formulations allowing for media with dielectric and conducting properties are emphasized. It is demonstrated that many problems in electromagnetism can be solved using the NASTRAN finite element code.

Several fundamental problems involving time harmonic solutions of Maxwell's equations with known analytic solutions are solved using NASTRAN to demonstrate convergence and mesh requirements. Mesh requirements are studied as a function of frequency, conductivity, and dielectric properties.

Applications in both low frequency and high frequency are highlighted. The low frequency problems demonstrate the ability to solve problems involving media inhomogeneity and unbounded domains. The high frequency applications demonstrate the ability to handle problems with large boundary to wavelength ratios.

INTRODUCTION

The Applied Mathematics Division at the David Taylor Research Center (DTRC) has begun developing methods using finite elements with NASTRAN to solve problems involving electromagnetic waves propagating in various media or scattered by objects in the field. This paper reports work supported by the Office of Naval Technology Exploratory Development Program, DTRC Project Manager, Dr. Bruce Hood.

The fundamental equations governing the propagation of electromagnetic waves are the Maxwell's equations. For many applications, the electric and magnetic field components satisfy the linear, damped wave or Helmholtz equation. While there are six field components in electromagnetic problems, for time harmonic fields, only three are independent. The equations are, therefore, similar to the Navier's equations governing an elastic solid.

In this paper, the several ways of formulating Maxwell's equations are presented. Formulations involving field vector components are compared with formulations involving potential functions. It is shown in the next section that it is possible to form analogies between Maxwell's equations and the Navier's equations. Standard finite element codes which solve the equations of elasticity (such as the NASTRAN code), therefore, with appropriate choices of material properties and boundary conditions, can be used to solve problems in electromagnetics.

In this paper, several example problems in electromagnetism are solved using elastic analogies and the NASTRAN finite element code. Examples of interest in low frequency applications and high frequency radar cross section applications are presented. The examples are all two dimensional; however, the analogies and the ability to solve electromagnetism problems with NASTRAN are not limited to two dimensional applications. All the applications use the IS2D8 element; however, any of the solid elements can be employed.

The problems of modeling point dipoles in both conducting and nonconducting media are studied in this paper. The accurate modeling of dipole sources is critical for applications involving electromagnetic waves. The results of these problems are compared with available analytic solutions. An important result is determining the mesh requirements needed to establish the dipole field accurately.

The mesh characteristics required for dipole modeling are employed for the study of the fields generated from a point dipole source located in sea water (which is a conducting, attenuating medium). Two frequencies representing the extremes of low frequency applications are presented. Of special interest is the study of the effect of a layer of ice on the solutions. While the example presented is somewhat idealized and limited, it should demonstrate to the reader the methodology required for the solution of low frequency problems.

Another example problem is the scattering of a plane wave by a conducting object. The problem of a circular cylinder in a plane wave field is solved and compared with the analytic solution. Excellent agreement is demonstrated using the IS2D8 element. Convergence of this element is quite superior to linear elements documented elsewhere. For this problem, both the electric field vector and the magnetic field vector equations are solved. This is analogous to the "sound soft" and "sound hard" scattering problems in acoustics.

Finally, the concluding section of this paper discusses areas where further development is required to solve some difficult, three dimensional problems. There is great potential in using standard finite element codes for the routine solution of electromagnetics problems.

MAXWELL'S EQUATIONS - FIELD STRENGTH AND POTENTIAL FORMULATION

Media in which electromagnetic waves travel often exhibit the properties of linearity, isotropy and homogeneity. This type of medium is called a linear, isotropic, and homogeneous (LIH) medium. For these media, the electric displacement vector, **D**, the magnetic field intensity,

H, the electric field intensity, **E**, the magnetic induction vector, **B** and the free current density, **J** are linearly related by the equations

$$\mathbf{D} = \epsilon \mathbf{E}; \qquad \mathbf{H} = \mathbf{B}/\mu; \qquad \mathbf{J} = \sigma \mathbf{E} \tag{1}$$

where ϵ is the permittivity , μ is the permeability and σ is the conductivity of the material. If we restrict our discussion to time harmonic fields at a single, arbitrary frequency, the field vectors **E** and **H** take the form

$$\begin{aligned} \mathbf{E} &= \mathbf{E}^{o} \exp(i\omega t) \\ \mathbf{H} &= \mathbf{H}^{o} \exp(i\omega t) \end{aligned} \tag{2}$$

where ω is the frequency and $i = \sqrt{-1}$. For time harmonic fields in an LIH medium, the governing equations for the spatial field variations can be derived from the Maxwell's equations and are given by [1]

$$\begin{aligned} \frac{\partial^2 E_i^o}{\partial x_j \partial x_j} + \frac{1}{c^2}\mu_r \epsilon_r^{\bullet} \omega^2 E_i^o &= 0 \\ \frac{\partial^2 H_i^o}{\partial x_j \partial x_j} + \frac{1}{c^2}\mu_r \epsilon_r^{\bullet} \omega^2 H_i^o &= 0 \end{aligned} \tag{3}$$

where

$$\epsilon_r^{\bullet} = \epsilon_r - \frac{i\sigma}{\omega \epsilon_o} \tag{4}$$

is the complex permittivity, c is the speed of light in the medium and ϵ_o is the permittivity of free space. The subscript, r, is not summed as is standard in the literature of electromagnetism [1] while the subscript i and j are summed in the standard cartesian tensor notation These equations are damped wave equations.

The governing equations given in (3) are the general equations to be solved for any problem in electromagnetics involving LIH media. It is important to note that these equations are uncoupled. The boundary conditions, however, may involve combinations of the field variables. The total problem, therefore, may be strongly coupled. This system, (3), represents six partial differential equations in the three components of **E** and of **H**. These variables, however, are not all independent. For time harmonic applications, only three of these equations are independent. For two dimensional time harmonic problems, only two of the components are independent. It is important, therefore, to insure that the problem under investigation is well posed. In practice, three dimensional electromagnetic problems are solved by solving for either **E** or **H** and calculating the other from the Maxwell equations. For special applications, one could choose two components of one field and one of the other. It is important to choose primary unknowns which are consistent with the available boundary conditions.

An alternative approach to the formulation of electromagnetic problems is to introduce potential quantities and derive governing equations for them from the Maxwell equations. A vector potential, **A** and a scalar potential ϕ are introduced by the relations

$$\begin{aligned} \mathbf{B} &= \nabla \times \mathbf{A} \\ \mathbf{E} &= -\nabla\phi - \frac{\partial \mathbf{A}}{\partial t} \end{aligned} \tag{5}$$

The potentials are not independent. They can be related to each other using the Lorentz gauge condition given by

$$\nabla \cdot \mathbf{A} + \epsilon\mu \frac{\partial \phi}{\partial t} = 0 \tag{6}$$

This is not the only possible gauge condition relating these quantities; however, it is the most widely employed [2].

If the relations in (5) are substituted into the Maxwell equations and the vector potential, **A**, is assumed to have a harmonic time variation given by

$$\mathbf{A} = \mathbf{A}^{\circ}\exp(i\omega t) \tag{7}$$

then the governing equation for **A** is

$$\frac{\partial^2 A_i^{\circ}}{\partial x_j \partial x_j} + \gamma \frac{\partial^2 A_j^{\circ}}{\partial x_i \partial x_j} + \kappa A_i^{\circ} = 0 \tag{8}$$

where

$$\kappa = \frac{\omega^2}{c^2} - i\omega\mu\sigma ; \qquad \gamma = -\frac{i\mu\sigma c^2}{\omega} \tag{9}$$

This system, (8), is also a damped wave-like equation with "shear coupling". The mixed partial derivative term comes from the conductance property of the medium. For nonconducting media, this equation reduces to an undamped Helmholtz equation.

NAVIER'S EQUATIONS AND ELASTIC ANALOGIES

For elastic bodies with time harmonic displacement response, the displacement vector for the steady state forced response (at frequency, ω) of the domain is given by

$$\mathbf{u} = \mathbf{u}^{\circ}\exp(i\omega t) \tag{10}$$

The governing equation for a damped, isotropic elastic media can be written as [3]

$$G\frac{\partial^2 u_i^\circ}{\partial x_j \partial x_j} + (\lambda+G)\frac{\partial^2 u_j^\circ}{\partial x_i \partial x_j} + (\rho\omega^2 - i\omega b)u_i^\circ = 0 \tag{11}$$

where G is the shear modulus, λ is the Lame constant, ρ is the mass density and b is the damping coefficient. An alternative form is

$$\frac{\partial^2 u_i^\circ}{\partial x_j \partial x_j} + H_1\frac{\partial^2 u_j^\circ}{\partial x_i \partial x_j} + H_2 u_i^\circ = 0 \tag{12}$$

where

$$H_1 = \frac{\lambda+G}{G}; \qquad H_2 = \frac{\rho\omega^2 - i\omega b}{G} \tag{13}$$

are complex material parameters. These are the equations which are solved by finite element codes designed for the solution of forced, harmonic elastic systems (such as NASTRAN).

It is desired to draw an analogy between the Navier equations and the Maxwell equations. This has been discussed in the literature previously for the scalar Helmholtz equation [4]. Following this approach, introduce the relation between Young's modulus, Y and the shear modulus G

$$Y = \alpha G = 2(1+\nu)G \tag{14}$$

where the Poisson's ratio, ν, is

$$\nu = \frac{\alpha}{2} \tag{15}$$

If the parameter α is chosen large enough so that

$$\alpha+1 \approx \alpha \tag{16}$$

then

$$H_1 \approx 0 \tag{17}$$

The Navier equations, under this choice of α, reduce to the Maxwell equations of (3). For most computers, a value of $\alpha = 10^{20}$ is usually sufficient [4]. The shear modulus, G, can be chosen arbitrarily.

If the problem of interest is two dimensional, the Navier equations must be reduced to the equations of either plane stress or plane strain. For plane stress, introduce the parameter, β, in the relation

$$Y = \beta G \tag{18}$$

where

$$\nu = \frac{\beta}{2} - 1 \tag{19}$$

If β is chosen so that

$$\beta + 1 \approx 1 \tag{20}$$

then, for the case of plane stress, [4]

$$H_1 \approx 0 \tag{21}$$

For scalar field problems on most computers, the choice of $\beta = 10^{-5}$ is sufficient [4]. The shear modulus, G, can still be chosen arbitrarily.

For either the two dimensional plane stress analogy or the three dimensional analogy, the complex electromagnetic material properties are related to the elastic properties through the equation

$$H_2 = \frac{\rho\omega^2 - i\omega b}{G} = \frac{1}{c^2}\mu_r \epsilon_r^{*} \omega^2 \tag{22}$$

The full Maxwell equations for an arbitrary LIH medium (two or three dimensional) can be solved by any finite element code which solves the Navier equations if the material properties are chosen appropriately and if the boundary conditions can be related to the applied forces and displacements. Boundary conditions will be discussed more rigorously in a later section.

An analogy can be formulated for the magnetic vector potential if the medium is nonconducting. In this case, the procedure is identical to the previous discussion as the mixed derivative terms do not appear. If the material is conducting, an analogy can be made if

$$H_1 = \gamma; \qquad H_2 = \kappa \tag{23}$$

This is possible if the elastic constants are complex. Since most structural codes do not permit complex material constants, the implementation would prove difficult. If one examines these material analogies, however, it can be seen that the required complex stiffness matrix can be formed by the sum of two real stiffness matrices multiplied by complex coefficients. NASTRAN can accomplish this by using DMAP instructions. The imaginary part of the stiffness matrix can be calculated in an analogy where

$$\omega\mu\sigma = \mathrm{H}_1; \qquad \frac{\mu\sigma c^2}{\omega} = \mathrm{H}_2 \tag{24}$$

This matrix can be saved, multiplied by $-i$, and added to the stiffness matrix for a problem with

$$\mathrm{H}_1 = 0; \qquad \mathrm{H}_2 = \frac{\omega^2}{c^2} \tag{25}$$

The new stiffness matrix will be the required matrix.

PIECEWISE HOMOGENEOUS MEDIA

In many applications, it is necessary to describe the electromagnetic fields which pass from one medium to another. Such problems are piecewise homogeneous. For problems with only dielectric materials (no conducting materials), this can be done by insuring all elements contain only one material and using different element material properties for the different media. The procedure is identical to solving problems where the density or elastic modulus varies from element to element.

For conducting media, the "viscous" damping coefficient needs to vary from element to element. This is not possible directly with the NASTRAN code. It is possible, through the use of DMAP statements, to simulate this with two matrix formulation runs. Form the mass matrix for the model with a mass density given by

$$\rho = \frac{G\mu_r\sigma}{c^2\epsilon_0} \quad \text{produces} \quad \mathbf{M}_1 \tag{26}$$

The mass density in each element can be different representing the different conducting media. It is desired to form a damping matrix, **B**, which has the element damping coefficients given by

$$b = \frac{G\mu_r\sigma}{c^2\epsilon_0} \quad \text{produces} \quad \mathbf{B} \tag{27}$$

This is accomplished if

$$\mathbf{B} = \mathbf{M}_1 \tag{28}$$

The first matrix formulation produces the $\mathbf{M}_1$ matrix. This can be written out and read in (using OUTPUT2/INPUT2) as the damping matrix in run two which now uses a mass density

$$\rho = \frac{G\epsilon_r\mu_r}{c^2} \quad \text{produces} \quad \mathbf{M} \tag{29}$$

to form the true analogous mass matrix. Each element can have different permittivities and permeabilities. The only requirement is that the material parameters are constant within an element.

Using the approach summarized above, multiple conducting media can be modeled with NASTRAN. An example of a layered media problem is presented in the following section. When this procedure is applied in NASTRAN, it is necessary to add a single damper element to the model with a zero damping coefficient. This will signal NASTRAN that the problem is fully complex and that the complex solver is required. Reading in the damping matrix is not sufficient for NASTRAN to choose the complex solver. If other dampers are present in the model, this is unnecessary.

TWO DIMENSIONAL EXAMPLE PROBLEMS

Several example problems are solved in this section to demonstrate the use of the analogies described previously. The problems presented range from very low frequency examples (at 1 Hz) to high frequency scattering examples (at 3 x 10^8 Hz). All the models employ the 8 node, quadratic, isoparametric quadrilateral element (IS2D8). These elements perform well for a variety of problems and yield accurate results for the problems with available analytic solutions.

EXAMPLE 1: A DIPOLE SOURCE IN FREE SPACE

As the first problem, the field produced by a two-dimensional point dipole in free space was computed to explore the use of analogies with NASTRAN. Information gained by computing the fields for this case will also be useful if fields in layered media need to be computed with a dipole source located in air. The finite element mesh used is shown in increasing detail in Figures 1, 2, and 3. Similar mesh configurations were used with two sizes of elements. For the larger elements the overall dimensions of the mesh (Figure 1) are $6 \cdot 10^8$ by $6 \cdot 10^8$ m. Thus each of the larger square elements in Figure 1 are 10^8 by 10^8 m, and there are three of these elements for each wavelength. The overall dimensions of the smaller mesh are $3 \cdot 10^8$ by $3 \cdot 10^8$ m. For this mesh there are six elements per wavelength. The relative dimensions of all elements in the two meshes are equal, so each is portrayed by the figures. The radial mesh in the lower left corner of Figure 1, which is graded down to ever smaller elements, contains the dipole source. In this section of the mesh, which has dimensions 10^8 by 10^8 m in the larger mesh ($5 \cdot 10^7$ by $5 \cdot 10^7$ m in the smaller), the elements are much smaller and the only consideration on the element size is to keep the aspect ratios within reasonable bounds (less than 1:8).

Boundary conditions are applied to the model to provide for wave absorption at the outer boundary, to apply symmetry conditions on the axes of symmetry, and to model the dipole source. The dipole boundary conditions are applied along the small circular boundary in the lower left-hand corner of Figure 3. Along the outer boundaries (upper and right sides in Figure 1), plane wave absorbing boundary conditions in the form of dashpots were applied. Along the axes of symmetry (lower and left sides in Figures 1, 2, and 3), symmetric boundary conditions were applied. The dipole source is modeled by imposing enforced values of the electric field for a dipole on the circular boundary sector in the lower left corner of Figure 3. The radius of this sector is 0.1 m. The complete solution for a two dimensional dipole can be found in [5]

The electric fields computed for this problem were compared with analytic solutions [5], and both meshes were found to produce reasonably accurate values. The amplitude and phase of the solution for the larger mesh are shown in Figures 4 and 5. and the amplitude and phase of the near-field solution are shown in Figures 6 and 7. The values plotted are the z-component of the electric field along a radial line 45 degrees from the lower axis. For the larger mesh, the error in the large square elements was on the order of 5 percent, and in the radial block the error was of the order of 1 percent. For this model, the region containing the radial elements is considered to be the region of interest, and the outside region is included only to model several wavelengths to provide for suitable wave absorbing boundaries. Therefore, in the region of interest, very good results were obtained.

For the larger mesh, two wave lengths were modeled before the absorbing boundary conditions are applied, and for the smaller mesh only one wave length was modeled. Decreasing the number of wave lengths modeled inside the boundary increased the error in the radial elements, the region of interest. The change in mesh size resulted in errors of 4 percent in both the square and radial elements in the smaller mesh. At the same time, the increase in the number of elements per wave length in the outer region slightly increased the accuracy there.

EXAMPLE 2: A DIPOLE SOURCE IN SEA WATER

Computing the field due to a dipole source in sea water was the first application to modeling electromagnetic fields in a conducting medium. As with the preceding problem, the region containing sea water was assumed to have infinite extent, so that comparisons could be made to an analytic solution [5]. Since for frequencies near one Hertz, sea water is a good conductor, the electromagnetic wave length, equal to 1581 m, is considerably shorter than $3 \cdot 10^8$ in free space. Therefore, the region modeled for this problem was correspondingly smaller than the region for the preceding problem. The finite element mesh used is shown in increasing detail in Figures 8, 9, and 10. The outer dimensions of this mesh are 5000 by 5000 m. In the outer region of Figure 8, the larger square elements are 250 m on a side, and the smaller square elements on the left side are 125 m on a side. The elements on the left were made smaller because this same mesh was to be used as part of the layered media problem, and the use of various element sizes allowed checking the performance of transitions from smaller to larger elements. The radial mesh in the lower left corner of Figure 8 contains the dipole source which is too small to be seen in this figure, but can be seen in Figure 10. This section of the mesh has dimensions 125 by 125 m. Here again the elements are much smaller, and the principal consideration is to keep the aspect ratios reasonable.

Again, boundary conditions are applied to provide for wave absorption on the outer boundary, to apply symmetry conditions on the axes of symmetry, and to model the dipole source. The dipole boundary conditions are applied along the small circular boundary in the lower left-hand corner of Figure 10. Along the outer boundaries (upper and right sides in Figure 8), plane wave absorbing dashpots were applied. Along the axes of symmetry (lower and left sides in Figures 8, 9, and 10), symmetric boundary conditions were applied. The dipole source is modeled by imposing enforced values of the electric field for a dipole on the circular boundary sector in the lower left corner of Figure 10. The radius of this sector is also 0.1 m.

The electric fields computed for this problem were compared with analytic solutions, and were found to produce accurate values. The amplitude and phase of the solution along the horizontal axis of symmetry are shown in Figures 11 and 12. The solution phase is plotted between -180 degrees and 180 degrees, therefore, an apparent discontinuity arises at radii at which the phase decreases past -180 degrees. The error in the solution was on the order of 1 percent everywhere. Again for this model, the region containing the radial elements is considered to be the region of interest. The outside region is included only to model enough of the medium to provide for wave absorbing boundaries. Therefore very good results were obtained in the region of interest. The amplitude and phase of the solution along the vertical axis of symmetry are shown in Figures 13 and 14. Excellent agreement with the analytic solution is demonstrated in this direction also.

EXAMPLE 3: MODELING A DIPOLE SOURCE IN A FINITE DEPTH OF SEA WATER

The problem under consideration is a dipole source located in a finite depth of sea water. The current modeling is limited to a two-dimensional line dipole. The general problem under consideration is shown in Figure 15. A two-dimensional dipole is located at a distance A beneath the surface of the sea water. The total depth of the sea water is H. The sea water may be covered by a layer of ice of thickness D. The air on top and the mud beneath the sea water are assumed to be infinite. The problem currently modeled assumes a sea depth of 250 m. The dipole source is located 125 m beneath the surface of the sea water. Models have been developed for sea water without ice and for sea water covered by 1 m of ice. The total mesh for the problem under consideration for a dipole source radiating at 1 hertz is shown in Figure 16.

The sea water is modeled for a total of 7500 m (approximately 30 skin depths) and then is terminated by a plane wave radiation boundary condition. The mud is modeled out to 16000 by 16000 m (which is approximately 20 skin depths). In the model shown, the air is also modeled out to 16000 by 16000 m. Both media are terminated by plane wave radiation boundary conditions. Since mud and sea water are attenuating media, the radiation boundary condition assumption is not expected to significantly influence the solution (this has been demonstrated for the case of a line dipole in an infinite region of sea water as discussed previously). For the air, however, it is often required to model a region on the order of several wave lengths. For air (at 1 hertz excitation), this corresponds to approximately 300,000,000 m. Results in air, however, are only of interest for distances less than 10,000 m from the source. The air, therefore, was modeled as far as the mud (for geometric symmetry) and absorbing boundary conditions were applied at the edge of the mesh.

Figure 17 shows a blowup of the entire sea water region. The transitioning mesh in the air and the sea water is shown. For the dipole in sea water, a mesh dimension of 125 by 125 m was demonstrated to predict accurate results. This is the dimension of the elements in the sea water as shown. In the air and mud, the elements are allowed to expand in a consistent manner to a final dimension of 1000 by 1000 m. The larger elements are permissible since the wavelength and attenuation distance in mud are larger than in sea water (the wavelength in mud is in the order of 5000 m and the skin depth in mud is on the order of 796 m). The transitioning is developed to insure that the element aspect ratios and interior angles remain within acceptable limits. Near the dipole source, a radially expanding mesh is employed as in the previous example. This mesh is sufficient to establish the near source field accurately. As in the previous example, the dipole is modeled as a small circular ring of nodes. On that ring of nodes, the analytic solution for a line dipole in an infinite medium of sea water is applied as a boundary condition. The model assumes, therefore, that close enough to the dipole, the ice, air and mud will have a negligible effect on the field variable solutions. The required inner mesh dimension will be determined by a convergence study. This parameter will be dependent upon the location of the source relative to the boundaries and the frequency of the source. The model described was modified to allow for a 1 m layer of ice. The resulting mesh is the same as the previous one except that between the sea water - air interface is a layer of elements 1 m thick which represent the ice. Since this dimension is small relative to the domain modeled, it is observable only on a blowup of the mesh.

Figures 18 and 19 show the amplitude and phase of the electric field component, E_Z, along the midline of the sea water. Solutions with and without ice are shown. A decaying field is observed with a characteristic knee in the solution. This occurs near the point where the phase crosses the zero line. This phenomena has also been observed experimentally [6]. The dropoff in the phase near the tail of the plot is probably due to the dashpot boundary condition. Figures 20 and 21 show the amplitude and phase of the E_Z component along the surface of the sea water. Qualitatively, the solution is similar to the midline solution. The amplitude does not, however, drop off as rapidly and the phase is shifted to a larger mean. It is interesting to note that at this low frequency the ice has negligible effect on the solution.

The same problem was studied for a higher frequency source at 1000 hz. The mesh employed is shown in Figure 22. The sea water region is modeled for 250 m by 250 m. This corresponds to about 50 skin depths. The mud is modeled for an additional 500 m corresponding to 20 skin depths in mud. The air is modeled out to 2000 additional meters. On all exterior boundaries, the dashpot absorbing conditions are employed.

The amplititude and phase of the E_Z component are shown in Figures 23 and 24 along the midline of the sea water. Again, solutions with and without ice are shown. The solution has a typical decaying amplitude with a sawtoothed phase characteristic. This is similar to the solution for a dipole in a conducting medium. It is interesting to note, however, that while the wavelength corresponds to the wavelength of the media, the decay is slower than for a dipole in infinite sea water. The amplitude is receiving significant contribution from the surface waves along the sea surface. Along the sea midline, little influence of the ice can be seen.

Figures 25 and 26 show the amplitude and phase of the E_z component along the surface of the sea water (with and without ice). The ice clearly has a significant influence on this solution. The amplitude without ice follows the amplitude with ice for about 10 skin depths of the sea water. The two solutions then change and the amplitude with ice is larger. Even though ice has a small conductivity (10^{-5} mhos), it acts as a wave guide keeping the surface wave of larger amplitude than without the ice. The phase, however, shows little difference with and without ice. The solutions are qualitatively similar with the exception that the ice guides the surface wave. Note that the deviation of the phase and the slight increase in amplitude toward the end of the plots is probably due to reflections from the dashpots.

EXAMPLE 4: PLANE WAVE SCATTERING FROM A RIGHT CIRCULAR CYLINDER

As a final example, consider the scattering of an incident plane wave by an infinite, perfectly conducting circular cylinder. The boundary condition on the cylinder is that the longitudinal component of the electric field must vanish on the surface of the cylinder and that the longitudinal component of the magnetic field must be normal to the surface. If the governing equations for the scattered wave only are considered, the boundary conditions for the scattered wave must remove the E_z component of the plane wave at the cylinder surface. In addition, the normal derivative of the H_z component of the plane wave must vanish at the surface of the cylinder. At infinity, the scattered wave must vanish. The problem considered is for a 1 m cylinder with an incident plane wave of 1 m wave length (the frequency, therefore, is 3 x 10^8 Hz).

The first mesh attempted employed eight elements in the azimuthal direction and quarter wavelength dimension in the radial direction. As is shown subsequently, this mesh performed adequately for the longitudinal component of the magnetic field but was not sufficient to accurately solve the longitudinal electric field problem. The mesh employed for the longitudinal electric field component is shown in Figure 27. The mesh was generated using the IDEAS [7] package. The design criterion was to generate a mesh as close to uniform in dimension as possible with an element size equal to one quarter of the incident wave length. The performance of this mesh was superior to that of a mesh with fixed radial dimensions of one quarter of a wavelength and aspect ratios within 1 to 5. The zero field condition was modeled with absorbing dashpots. For this case, a cylindrical wave condition would be superior due to the geometry of the problem. This was compared with the simple dashpot condition for the magnetic field solution.

Figure 28 is a plot of the normalized amplitude of the scattered electric field intensity on the forward scattering side of the cylinder. The overall agreement is quite good. The maximum error is less than 5% compared with the analytic solution. Very near the cylinder, however, the largest deviation is observed. Indeed it is only in this region where the error is larger than 1%. Figure 29 is a blowup of this region. The analytic solution flattens near the cylinder while the finite element solution demonstrates a sharp dip. There is significant ripple in the solution in this region which may indicate reflection problems between the cylinder and the dashpot. A blowup of the analytic solution is shown in Figure 30. The region which appeared flat in Figure 29 has a slight dip as demonstrated in Figure 30. The finite element solution exaggerates this dip. Since the elements are on the order of 0.25 m (one quarter of a wavelength), it is evident from Figure 30 that this mesh density would be insufficient to totally reproduce this phenomena. Overall, however, the solution is quite good.

Figure 31 shows a plot of the phase of the forward scattered field. The finite element results are almost identical to the analytic solution. This demonstrates that while small amplitude errors may be introduced into the solution, the general character of the waves are accurately predicted by the finite element solution. Figure 32 shows the normalized amplitude of the scattered electric field on the back side of the cylinder. The finite element results agree very well with the analytic solution. All errors are bounded by 1% even near the dashpot boundary condition. Figure 33 shows a plot of the phase of the electric field on the backscattering side of the cylinder. Again, excellent agreement is seen. The sawtoothed phase characteristic is accurately predicted and the ramping behavior is accurate.

For this example, the longitudinal component of the magnetic field vector was also resolved. The longitudinal components of the E and H fields are the only independent components for the two dimensional applications. Figure 33 shows the amplitude of the H_z component as a function of distance away from the cylinder along the back scattering side. The finite element solution is only negligibly different from the analytic solution. Figure 34 shows the amplitude along a radial line at 112.5 degrees from the incident wave. This represents the worst case and yet the two solutions agree quite well. It should be noted that the total amplitude along this line is quite small. It is remarkable that the solution is this accurate. In addition, the finite element results quite accurately capture the spiked dip in the solution even though only four elements per wavelength were employed.

An important observation is that accurate solutions were generated with approximately four IS2D8 elements per wavelength. This problem has been solved previously with linear quadrilateral elements [8]. In that study, ten elements per wavelength were required necessitating a significantly greater number of degrees of freedom to achieve an accurate solution.

CONCLUDING REMARKS

Maxwell's equations were solved for a variety of example problems in two dimensions. An interesting outcome of the low frequency examples was the ability to predict the wave guide effects of the ice in the layered media problem and the knee response in the amplitude. In addition, this problem demonstrated that relatively complicated problems can be solved by routine methodology. This conclusion should also hold for three dimensional applications.

The scattering example demonstrates the ability to handle high frequency applications. The conclusion that only four IS2D8 elements are required per wavelength indicates that considerable economy should be realized by using quadratic elements for harmonic response applications. This conclusion should be valid for structural and acoustic applications as well as for electromagnetic applications.

Since the two dimensional problems exhibit totally uncoupled boundary conditions, the solution of two dimensional problems in electromagnetics is the same as solution of the scalar wave equation. In three dimensions, this is not the case. The boundary conditions encountered are often coupled. This poses a problem for certain situations.

A common three dimensional boundary condition is the perfect conductor condition of zero tangential E and normal H. For high frequency applications, this is not a problem because several skin depths of the conductor can be modeled easily since this dimension will be small relative to the conductor's size. The conductor will damp out and absorb the waves appropriately. This indicates that radar cross section problems in three dimensions can be handled by elastic analogies.

For low frequency applications, the presence of a conductor is not as easy to deal with since the skin depth is often large relative to the size of the conductor. The vanishing of the tangential E field can be handled by multipoint constraints (MPCs). The vanishing of the normal H field is not as trivially solved. Methodologies for enforcing this condition are under investigation. It may be possible to extend the concept of MPCs to include linear combinations of first partial derivatives. This would solve the problem.

The other major problem to be addressed is the fact that many electromagnetic problems are exterior problems. They involve either extremely large or infinite domains. The solution is of interest, however, only in a small domain. It is necessary, therefore, to reduce the modeled domain and to implement a boundary condition which accounts for the remaining media. In this paper, the simple plane wave condition was employed. While this works, often large domains must be modeled. Other conditions have been explored; however, additional research is required. Infinite elements (employed for some limited scalar applications [9]) hold promise. These are currently being investigated also.

The remainder of the boundary conditions encountered in most applications can be handled trivially with elastic finite element codes like NASTRAN. This paper has demonstrated the ability to handle two dimensional problems and has provided the formulation for three dimensional problems. When absorbing boundary conditions become available and the zero normality condition for the H field is developed, it will be possible to solve virtually all problems in electromagnetics (which adhere to the assumptions of Maxwell's equations) with elastic finite element codes like NASTRAN.

REFERENCES

1. P. Lorrain and D. Corson, *Electromagnetic Fields and Waves,* W.H. Freeman and Co., San Francisco, 1970.

2. J.D. Jackson, *Classical Electrodynamics,* John Wiley and Sons, New York, 1975.

3. L.M. Brekhovskikh, *Waves in Layered Media,* Academic Press, New York, 1960.

4. G.C. Everstine, "Structural Analogies for Scalar Field Problems," *International Journal for Numerical Methods in Engineering,* Vol. 17, No. 3, pp. 471-476, 1981.

5. D.S. Jones, *The Theory of Electromgnetism,* Pergamon Press, Oxford, 1964.

6. R. King, G.S. Smith, M. Owens and T.T. Wu, *Antennas in Matter,* The MIT Press, Cambridge, Mass., 1981.

7. "SUPERTAB Engineering Analysis Pre- and Post-Processing User Guide and Reference Manual," Structural Dynamics Research Corporation, Milford, Ohio, 1976.

8. A.B. Bruno and J.R. Brauer, "Scattering from an Infinite, Finitely Conducting Cylinder by Finite Element Analysis," *Journal Of Applied Physics,* Vol. 63, No. 8, pp. 3200-3202, 1988.

9. O.C. Zienkiewicz, K. Bando, P. Bettess, C. Emson and T.C. Chiam, "Mapped Infinite Elements For Exterior Wave Problems," *International Journal for Numerical Methods in Engineering,* Vol. 21, pp. 1229-1251, 1985.

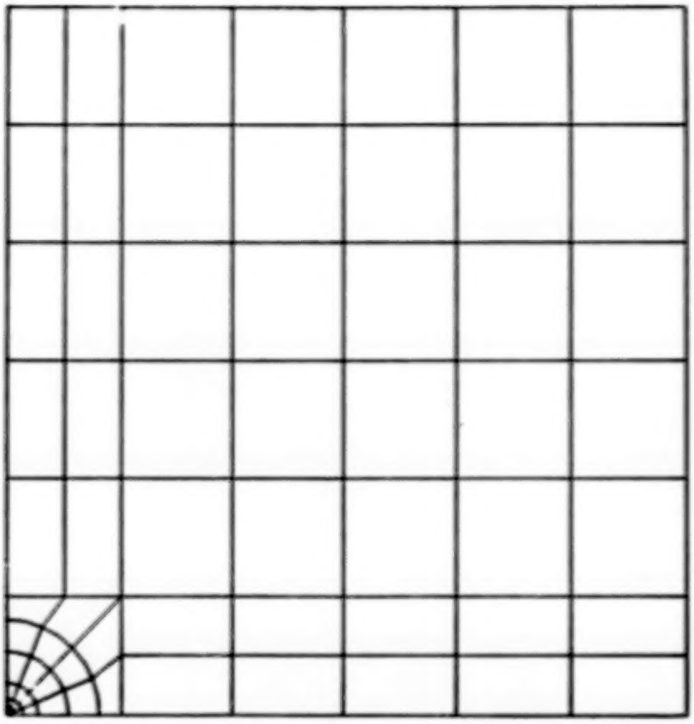


Fig. 1. FINITE ELEMENT MODEL FOR POINT DIPOLE IN FREE SPACE

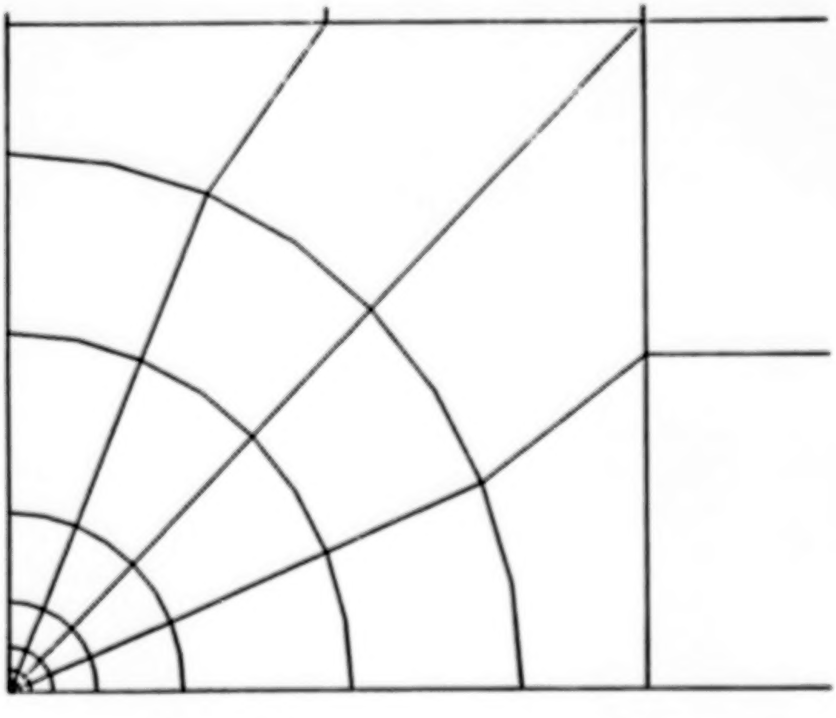


Fig. 2. RADIAL DIPOLE MODEL

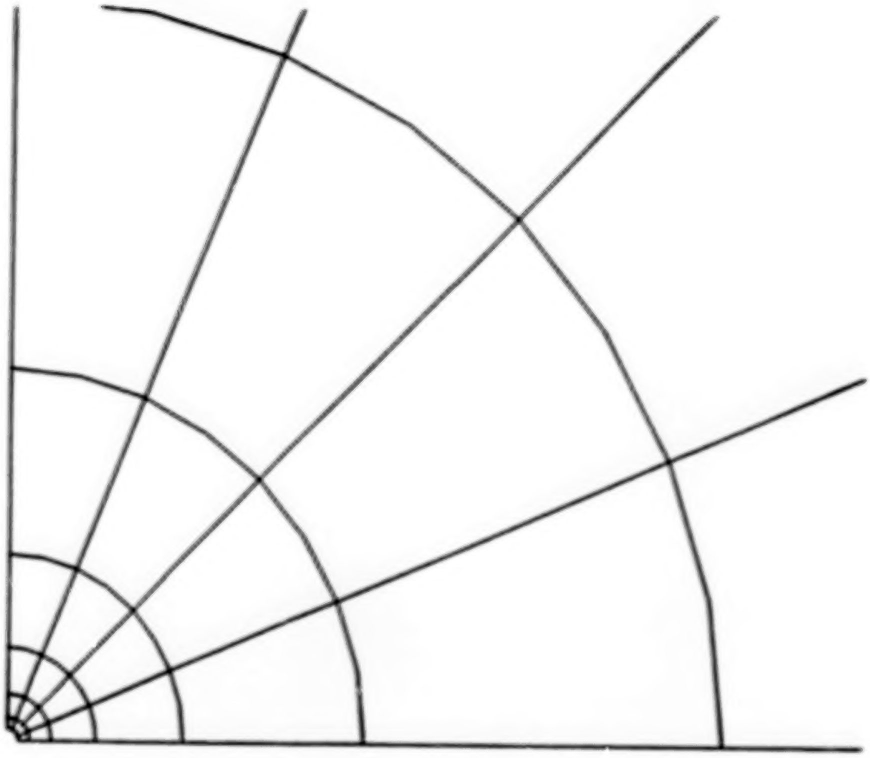


Fig. 3. NEAR DIPOLE MODELING

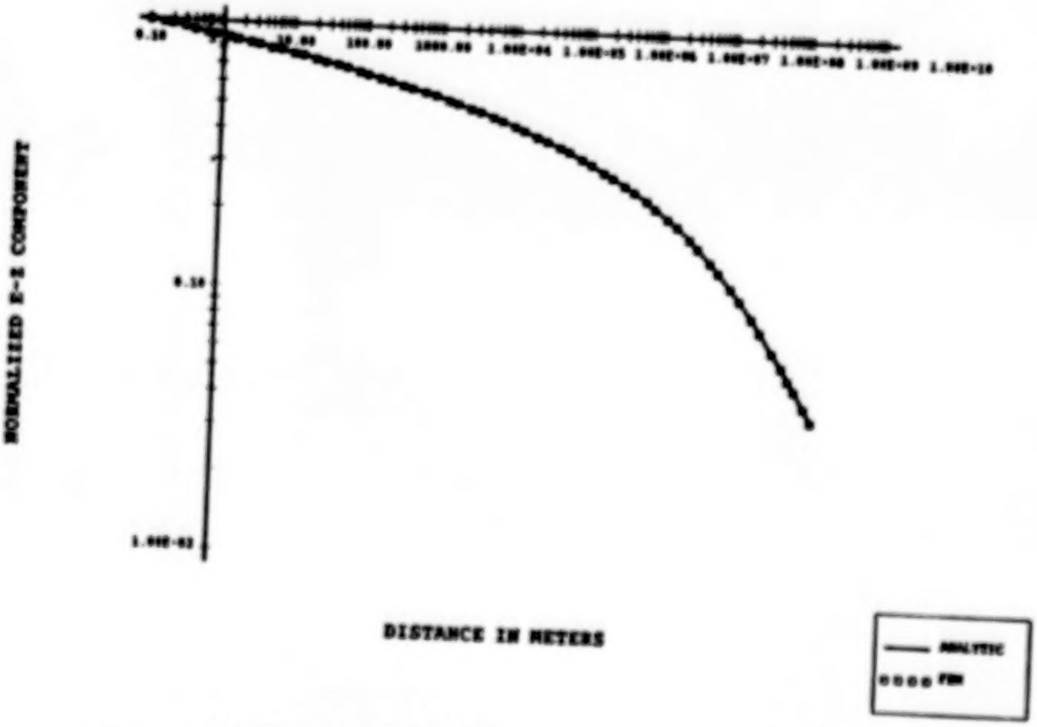


Fig. 4. AMPLITUDE FOR POINT DIPOLE IN FREE SPACE

BEST COPY
AVAILABLE

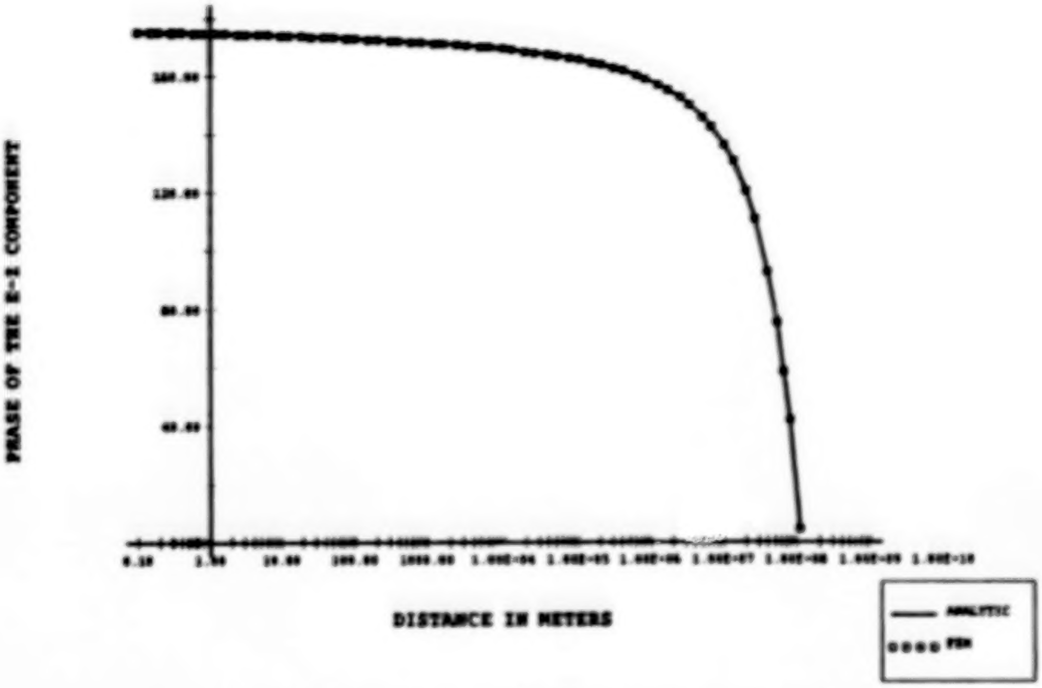


Fig. 5. PHASE FOR POINT DIPOLE IN FREE SPACE

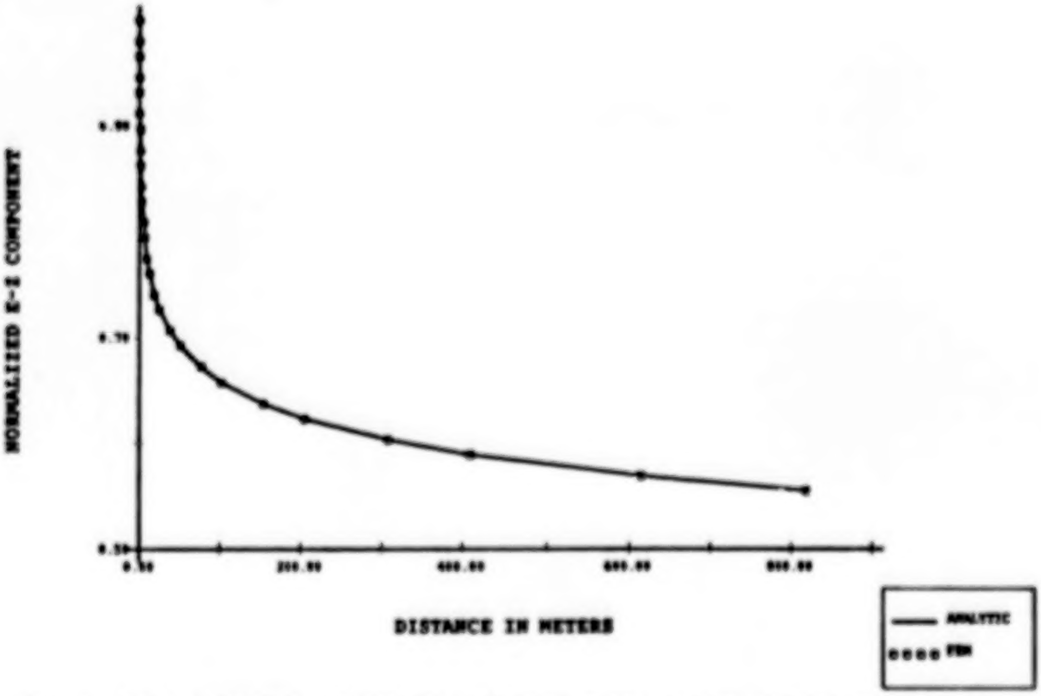


Fig. 6. NEAR FIELD AMPLITUDE FOR POINT DIPOLE IN FREE SPACE

BEST COPY AVAILABLE

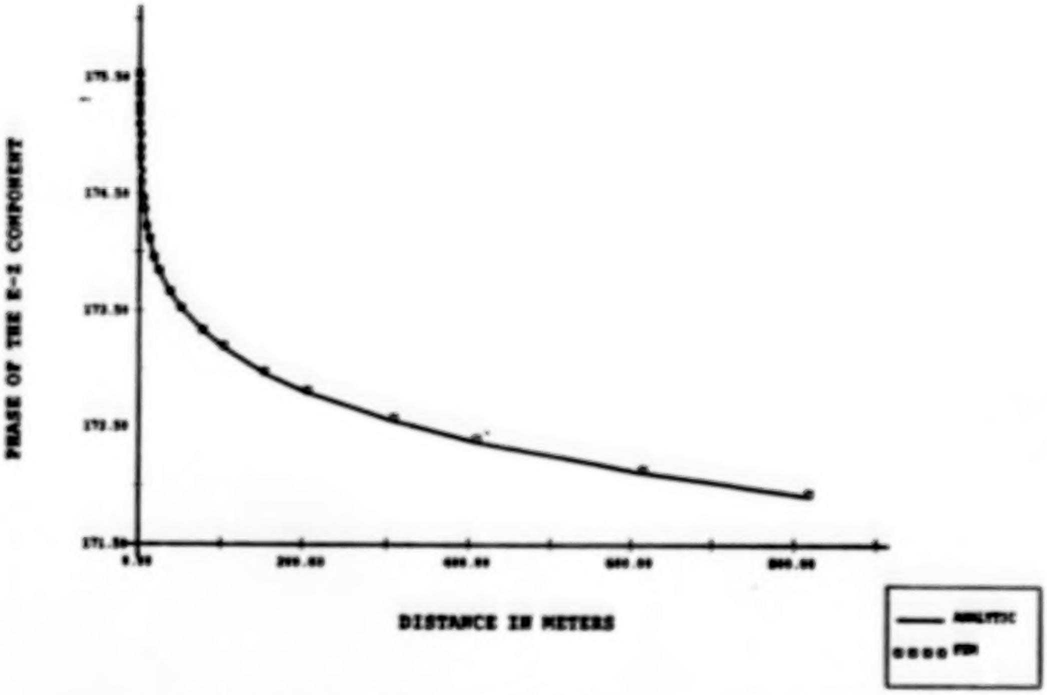


Fig. 7. NEAR FIELD PHASE FOR POINT DIPOLE IN FREE SPACE

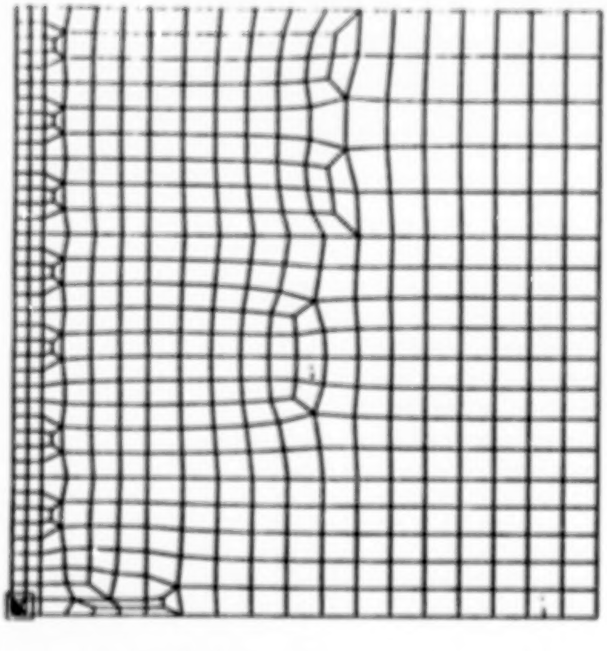


Fig. 8. FINITE ELEMENT MODEL FOR POINT DIPOLE IN SEA WATER

BEST COPY
AVAILABLE

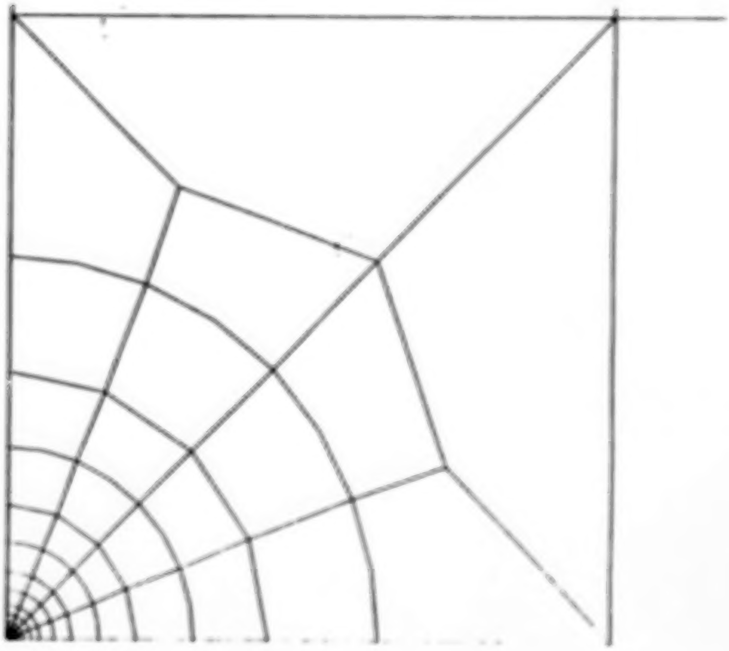


Fig. 9. RADIAL DIPOLE MODEL IN SEA WATER

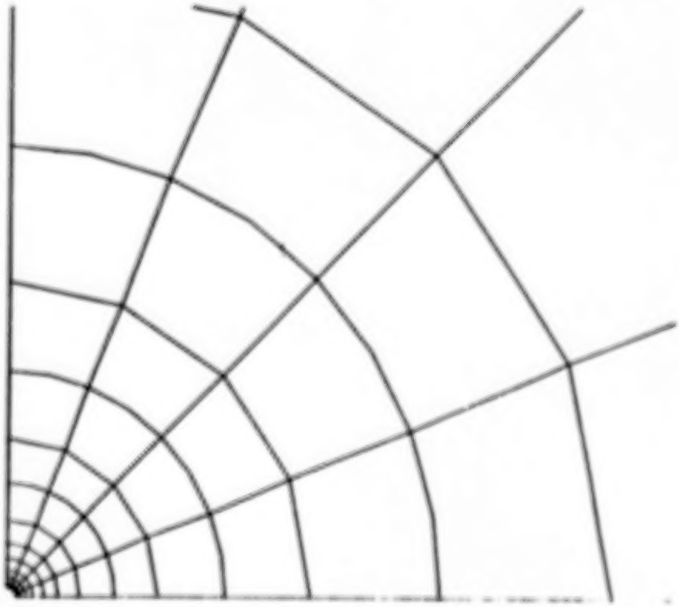


Fig. 10. NEAR DIPOLE MODEL IN SEA WATER

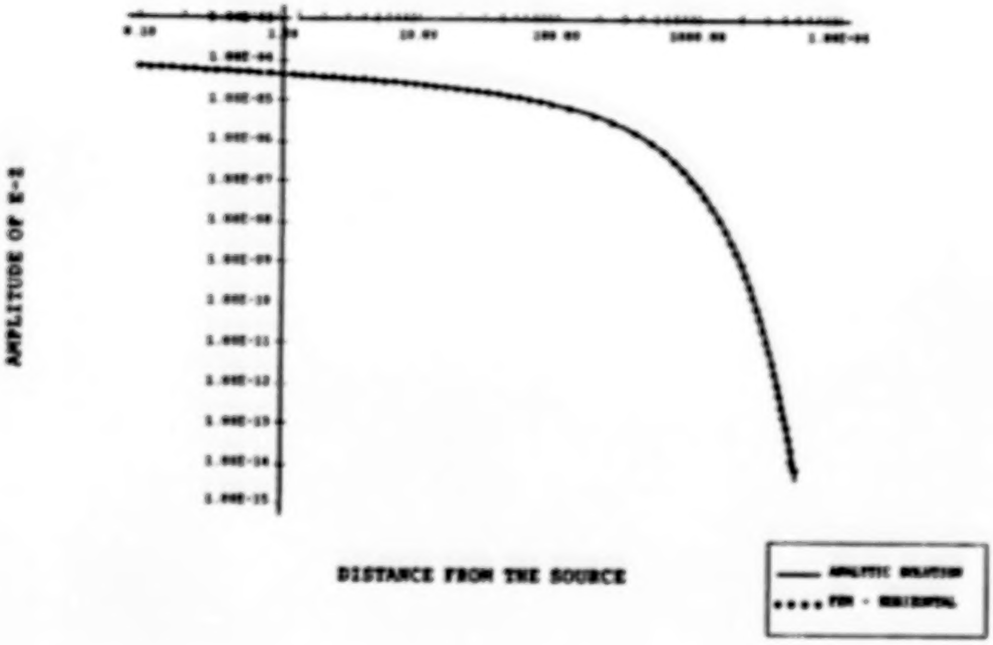


Fig. 11. AMPLITUDE ALONG HORIZONTAL SYMMETRY AXIS FOR POINT DIPOLE IN SEA WATER

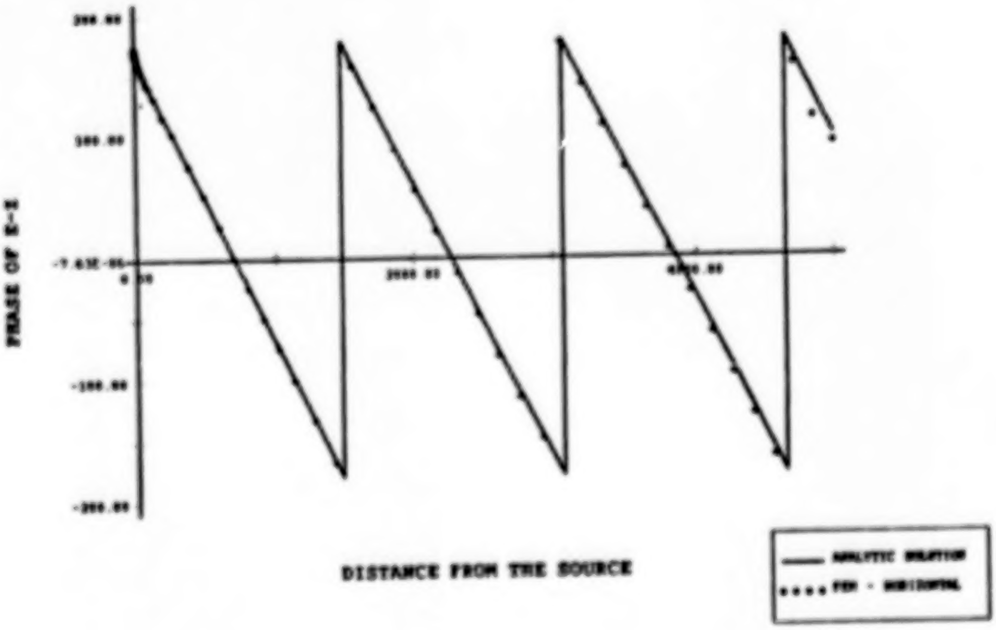


Fig. 12. PHASE ALONG HORIZONTAL SYMMETRY AXIS FOR POINT DIPOLE IN SEA WATER

BEST COPY AVAILABLE

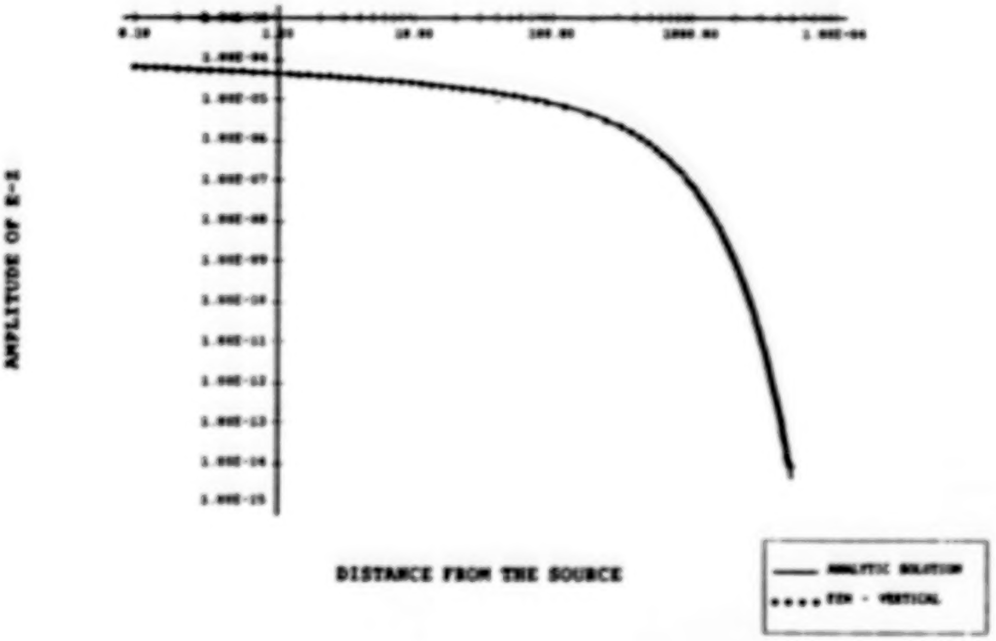


Fig. 13. AMPLITUDE ALONG VERTICAL SYMMETRY AXIS FOR POINT DIPOLE IN SEA WATER

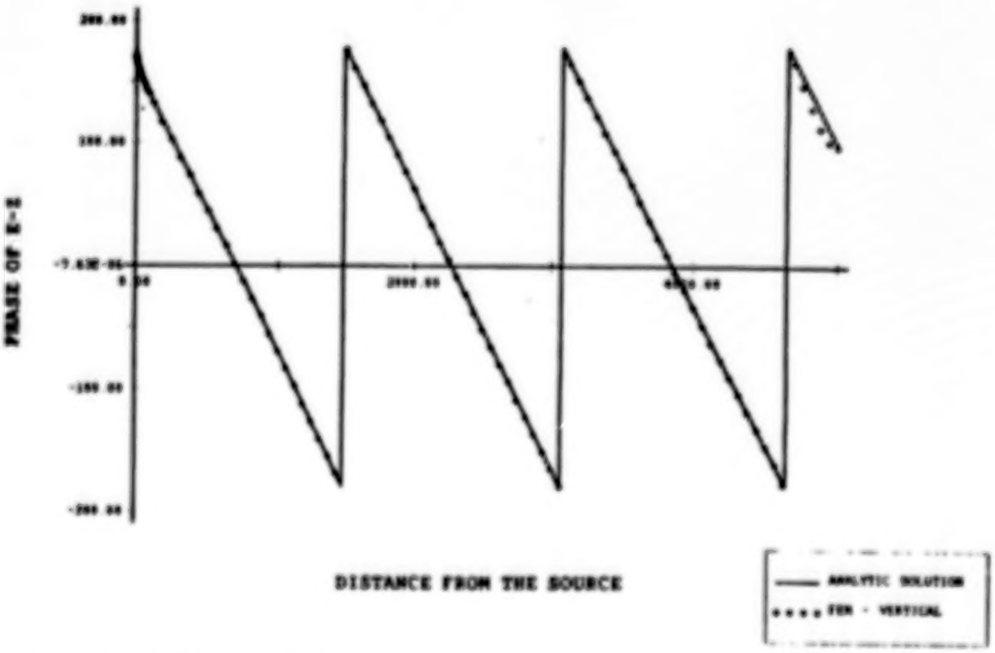


Fig. 14. PHASE ALONG VERTICAL SYMMETRY AXIS FOR POINT DIPOLE IN SEA WATER

BEST COPY AVAILABLE

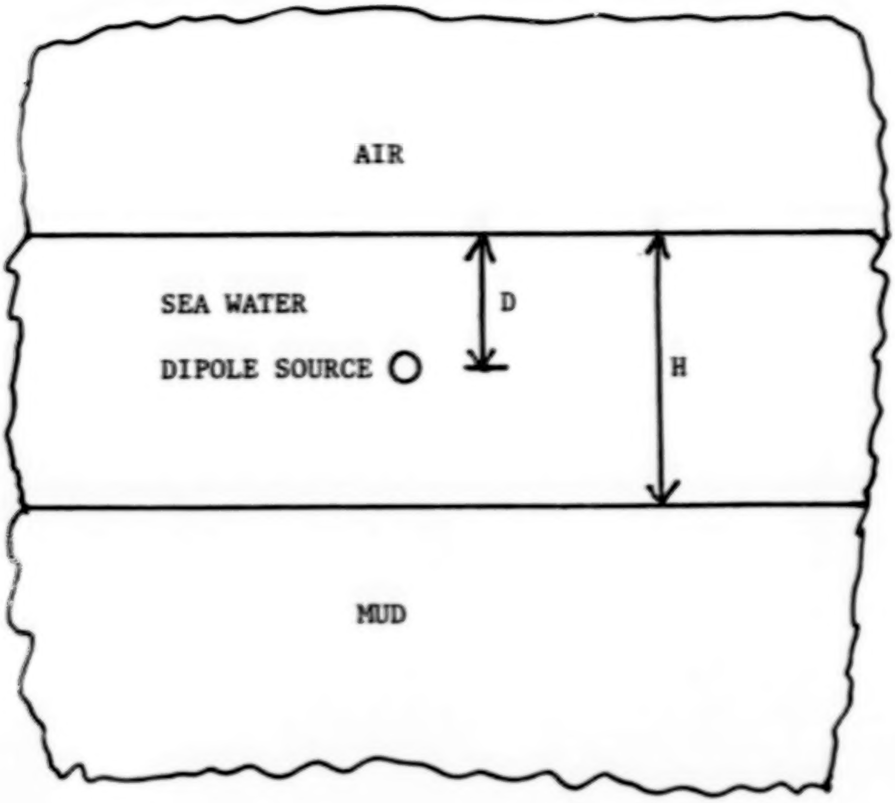

AIR
SEA WATER
D
DIPOLE SOURCE
H
MUD

Fig. 15. POINT DIPOLE IN SEA WATER WITH AIR, ICE AND MUD

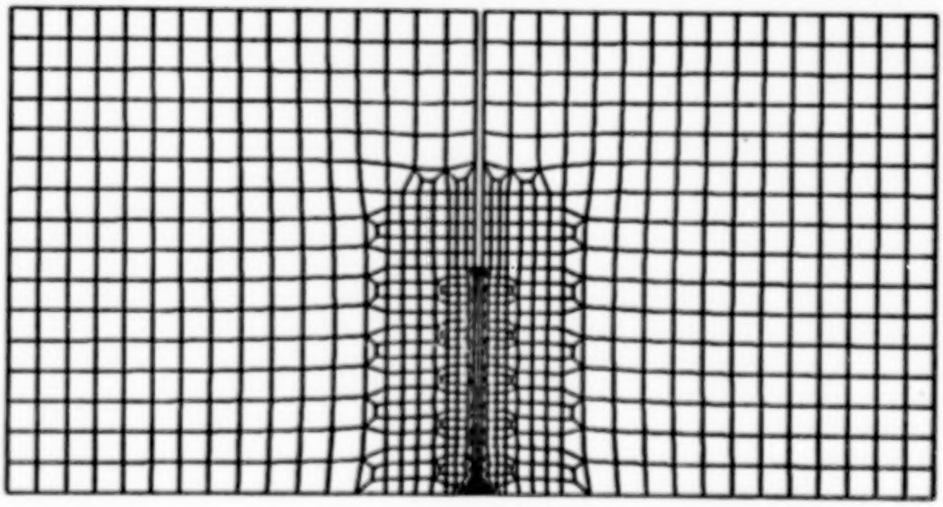


Fig. 16. FINITE ELEMENT MODEL FOR DIPOLE IN SEA WATER WITH AIR, MUD AND ICE

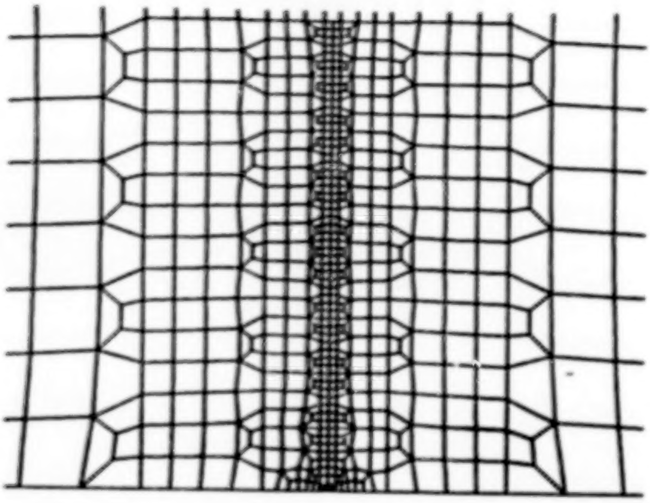


Fig. 17. BLOWUP OF SEA WATER REGION

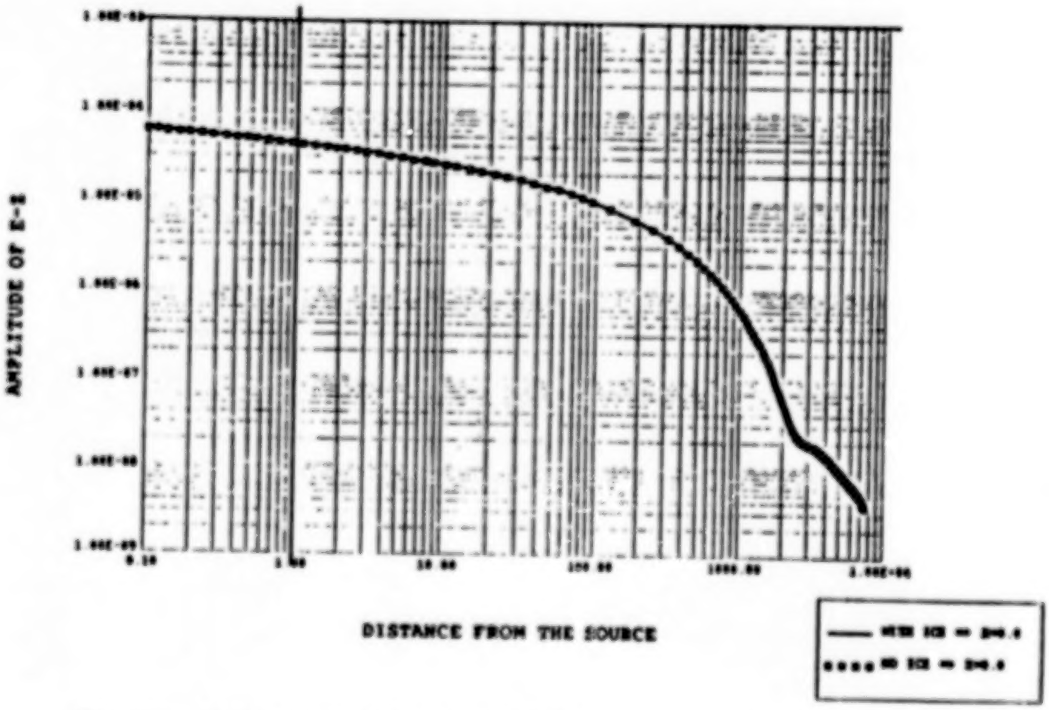


Fig. 18. AMPLITUDE OF E-Z ALONG SEA MIDLINE - 1 HZ.

BEST COPY
AVAILABLE

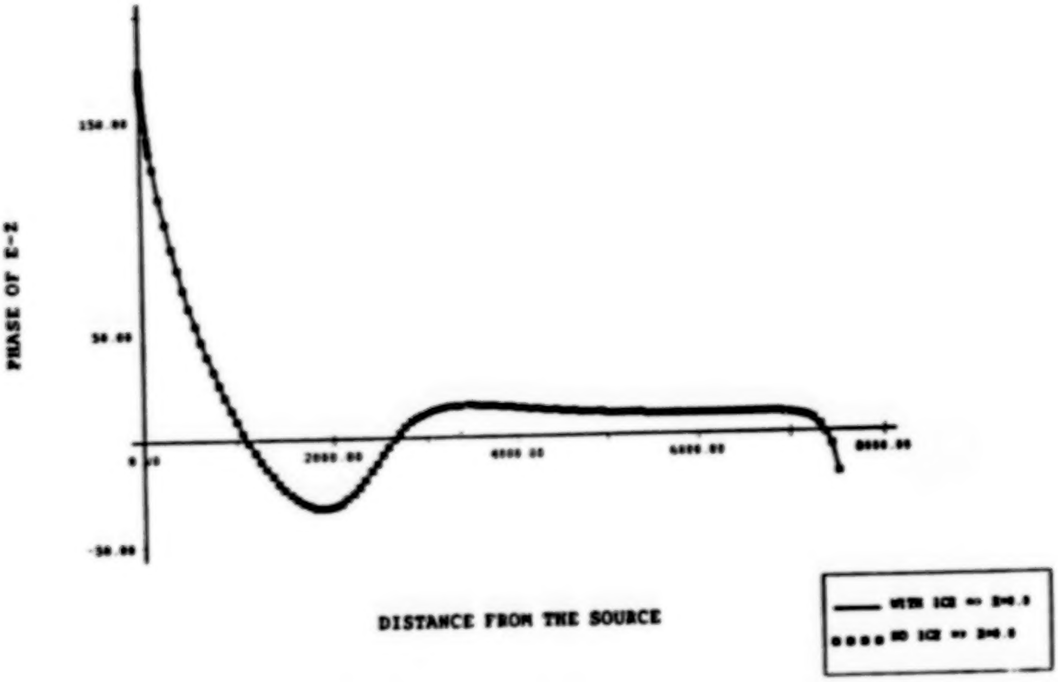


Fig. 19. PHASE OF E-Z ALONG SEA MIDLINE - 1 HZ.

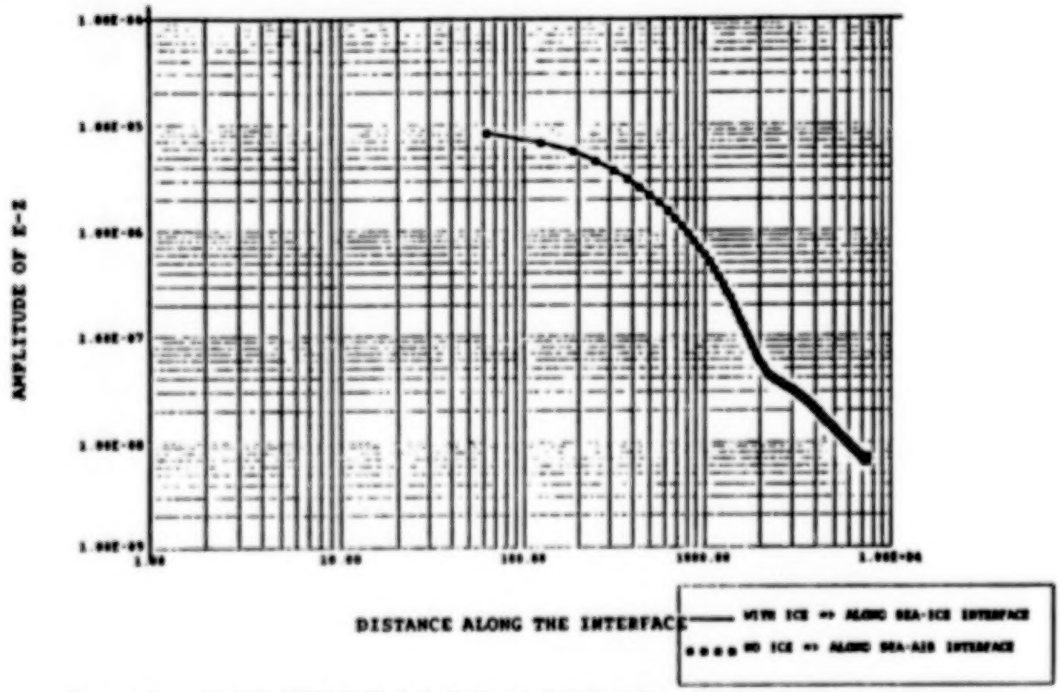


Fig. 20. AMPLITUDE OF E-Z ALONG SEA SURFACE - 1 HZ.

BEST COPY
AVAILABLE

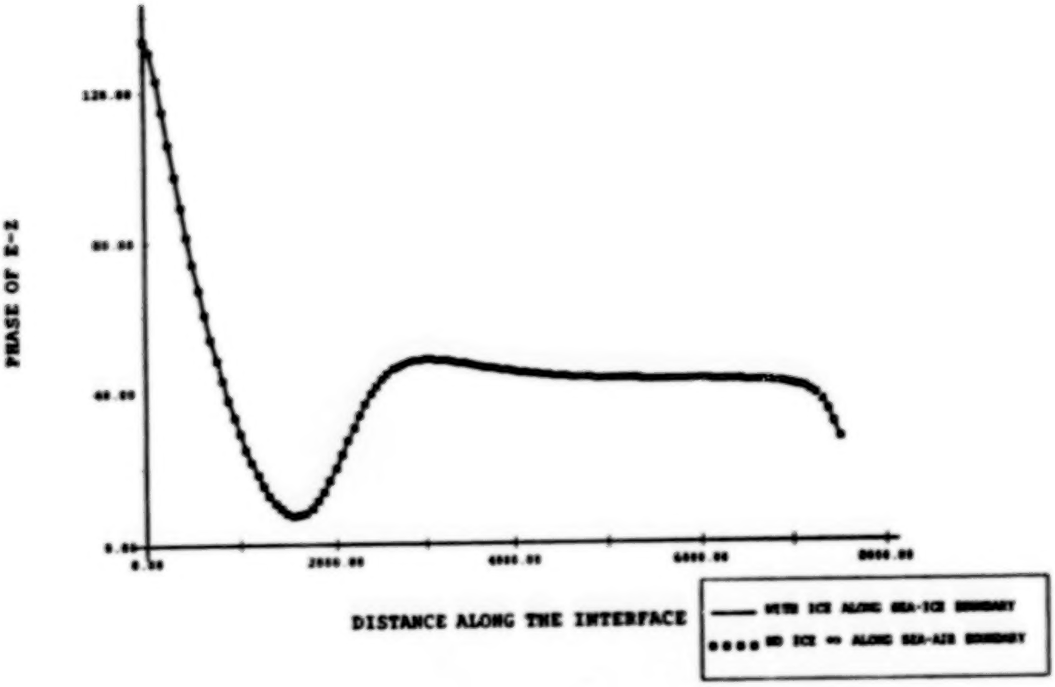


Fig. 21. PHASE OF E-Z ALONG SEA SURFACE - 1 HZ.

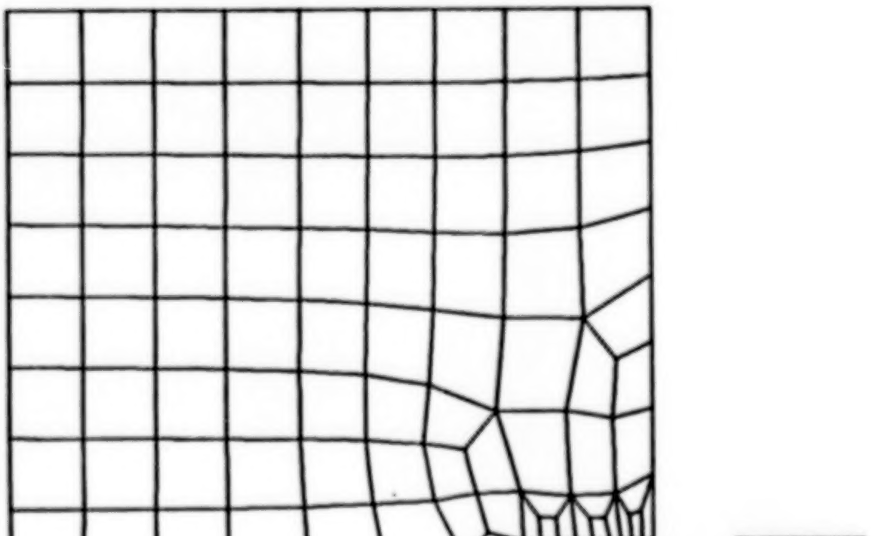


Fig. 22. FINITE ELEMENT MESH FOR 1000 HZ. SOURCE

BEST COPY
AVAILABLE

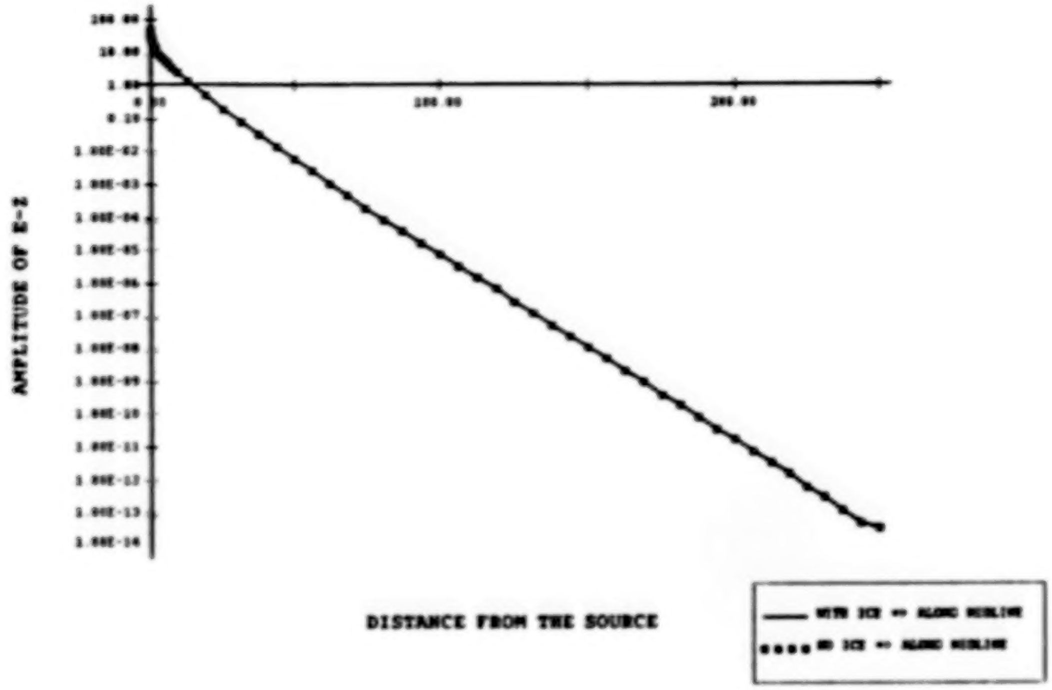


Fig. 23. AMPLITUDE OF E-Z ALONG SEA MIDLINE - 1000 HZ.

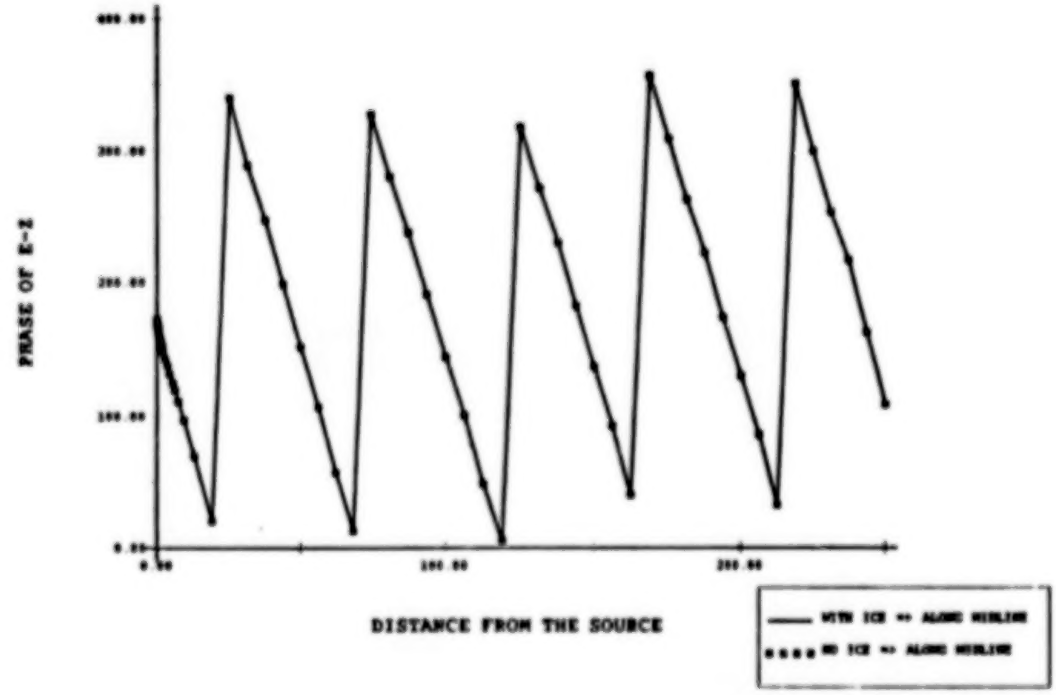


Fig. 24. PHASE OF E-Z ALONG SEA MIDLINE - 1000 HZ.

BEST COPY
AVAILABLE

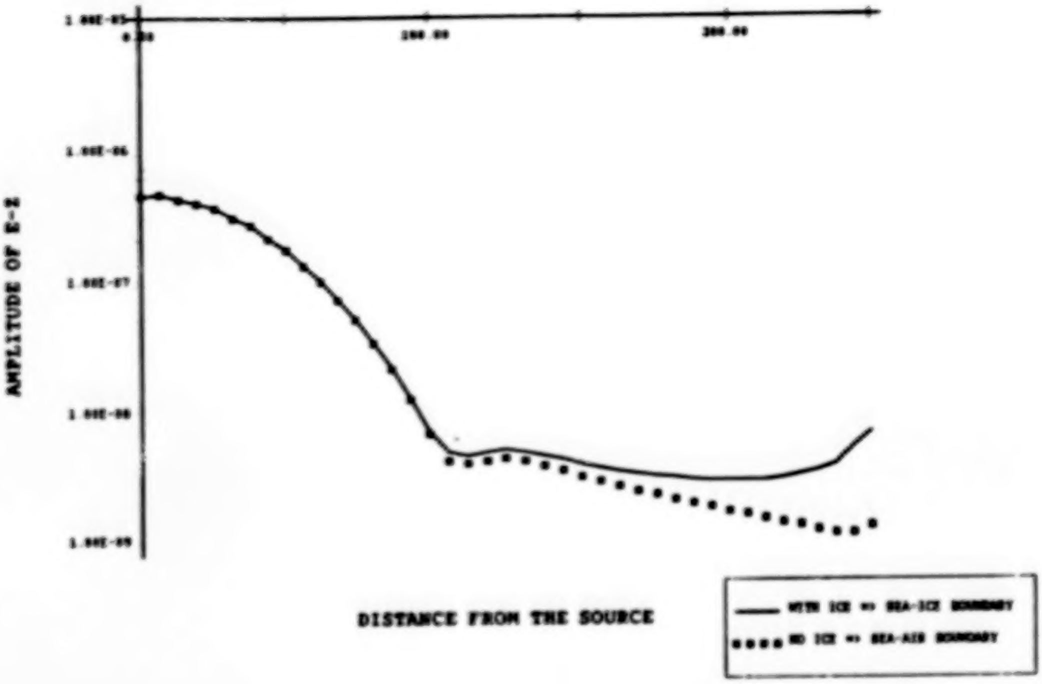


Fig. 25. AMPLITUDE OF E-Z ALONG SEA SURFACE - 1000 HZ.

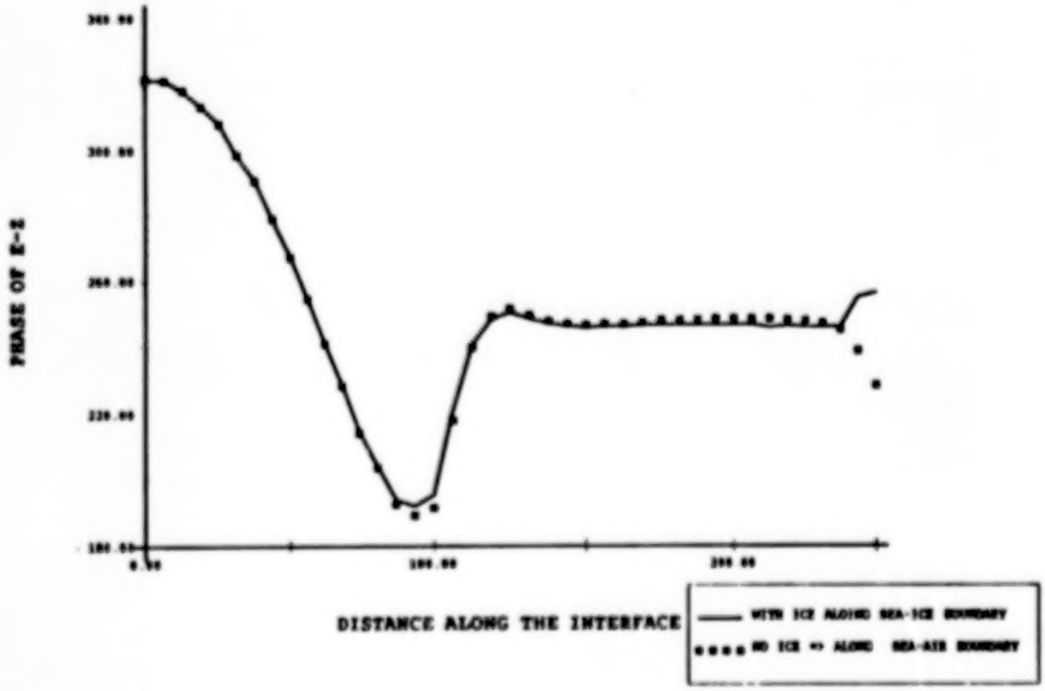


Fig. 26. PHASE OF E-Z ALONG SEA SURFACE - 1000 HZ.

BEST COPY
AVAILABLE

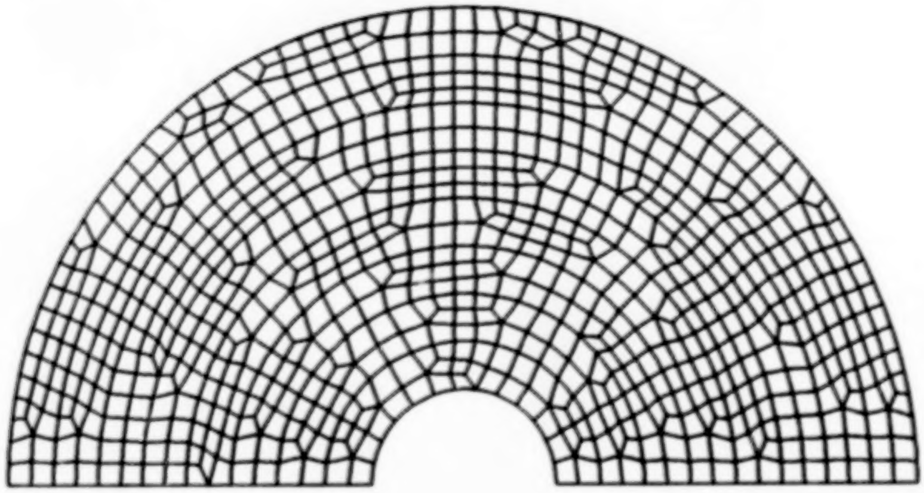


Fig. 27. FINITE ELEMENT MESH FOR SCATTERING ABOUT CIRCULAR CYLINDER

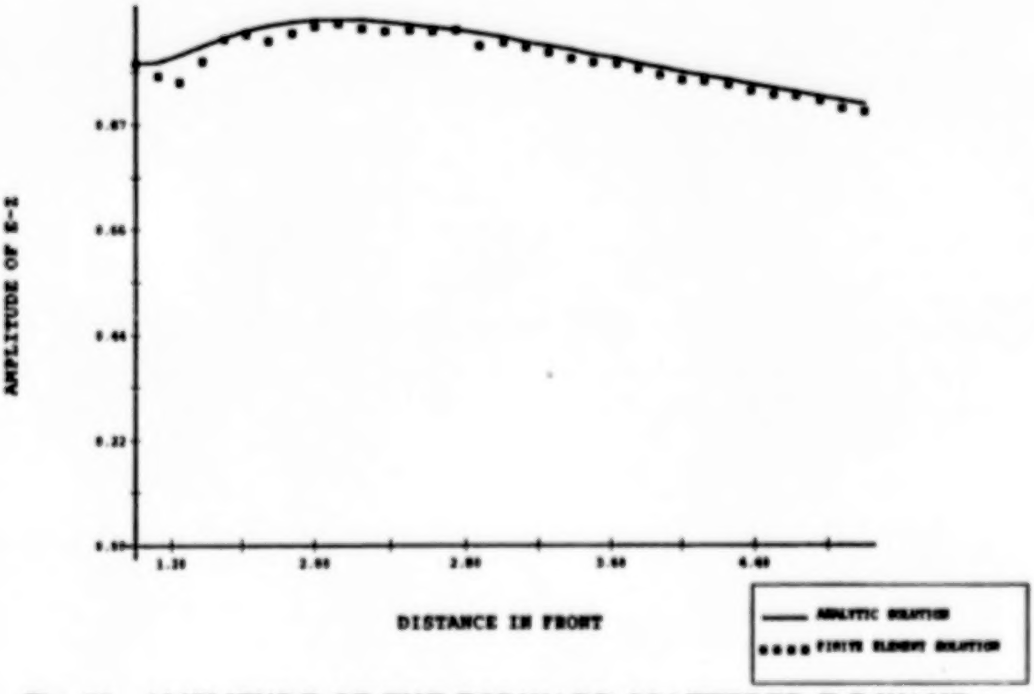


Fig. 28. AMPLITUDE OF THE FORWARD SCATTERED E-Z WAVE

BEST COPY
AVAILABLE

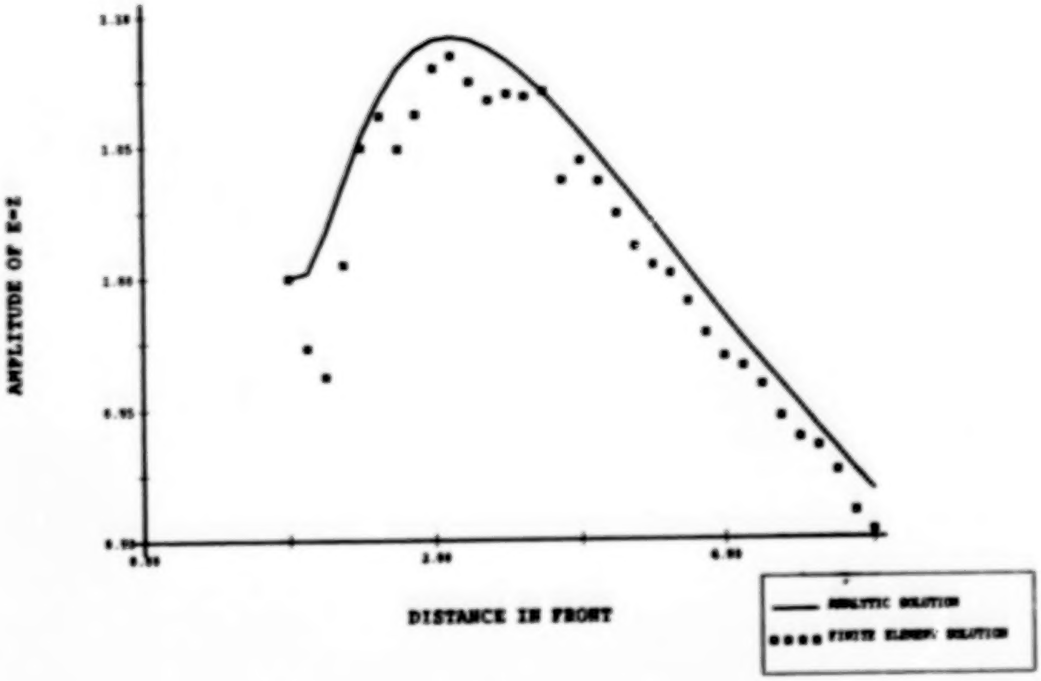


Fig. 29. BLOWUP OF THE SCATTER IN THE E-Z AMPLITUDE SOLUTION

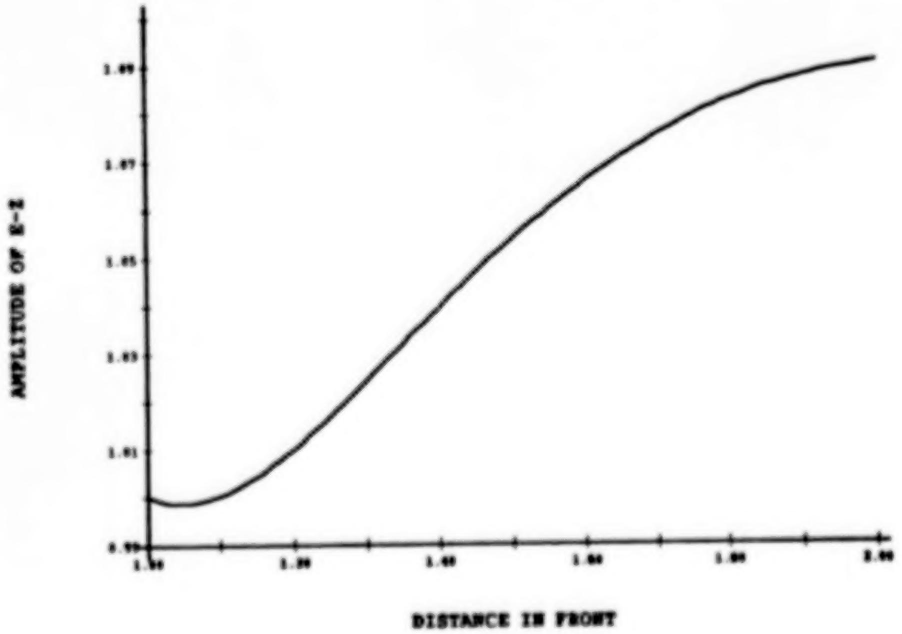


Fig. 30. NEAR CYLINDER BEHAVIOR IN THE ANALYTIC SOLUTION

BEST COPY
AVAILABLE

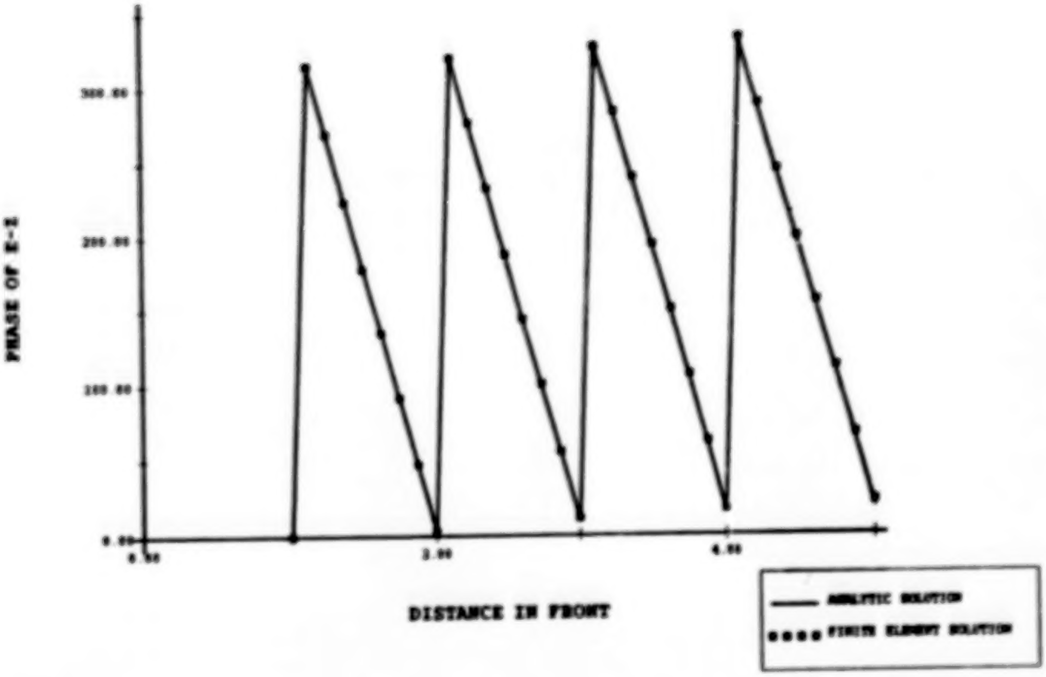


Fig. 31. PHASE OF THE FORWARD SCATTERED E-Z WAVE

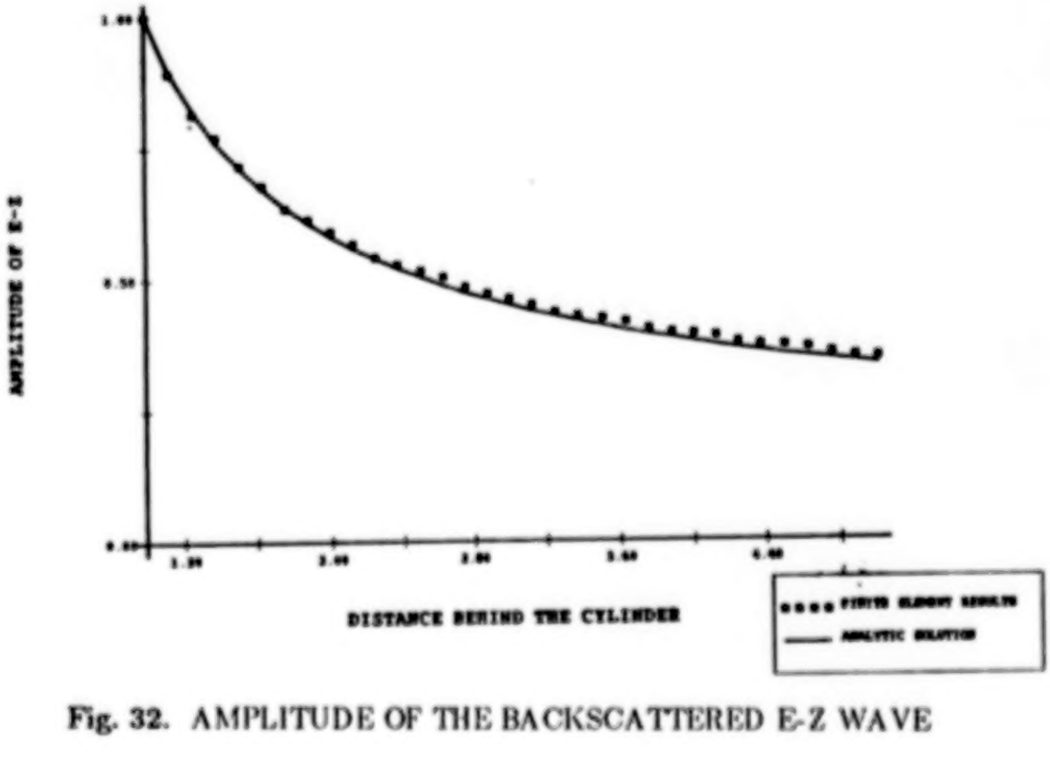


Fig. 32. AMPLITUDE OF THE BACKSCATTERED E-Z WAVE

BEST COPY AVAILABLE

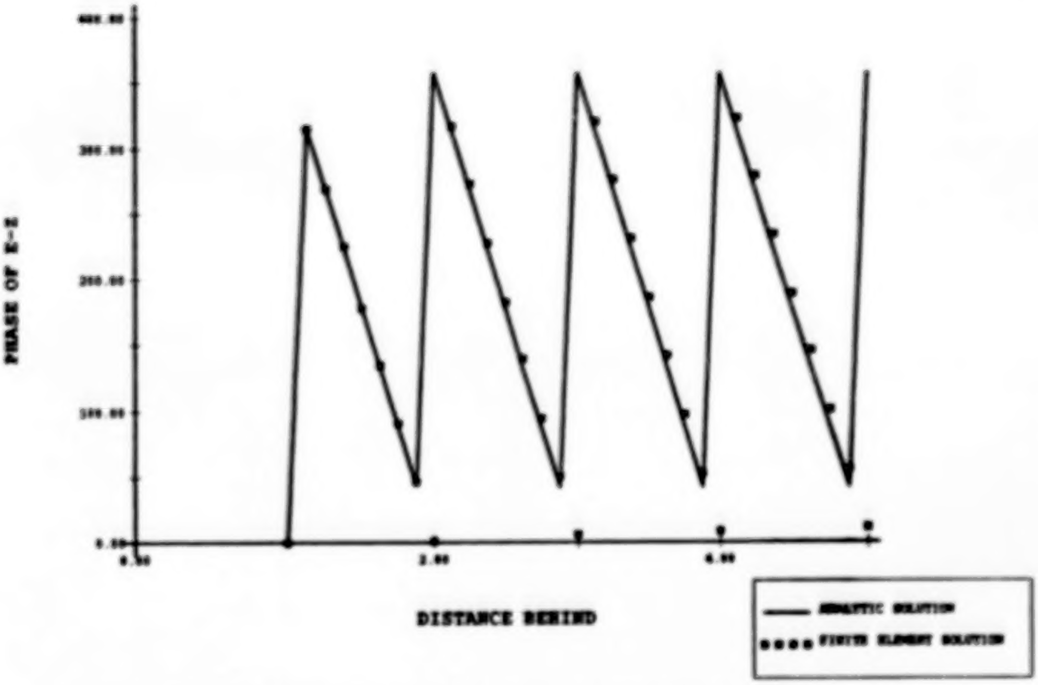


Fig. 33. PHASE OF THE BACK SCATTERED E-Z WAVE

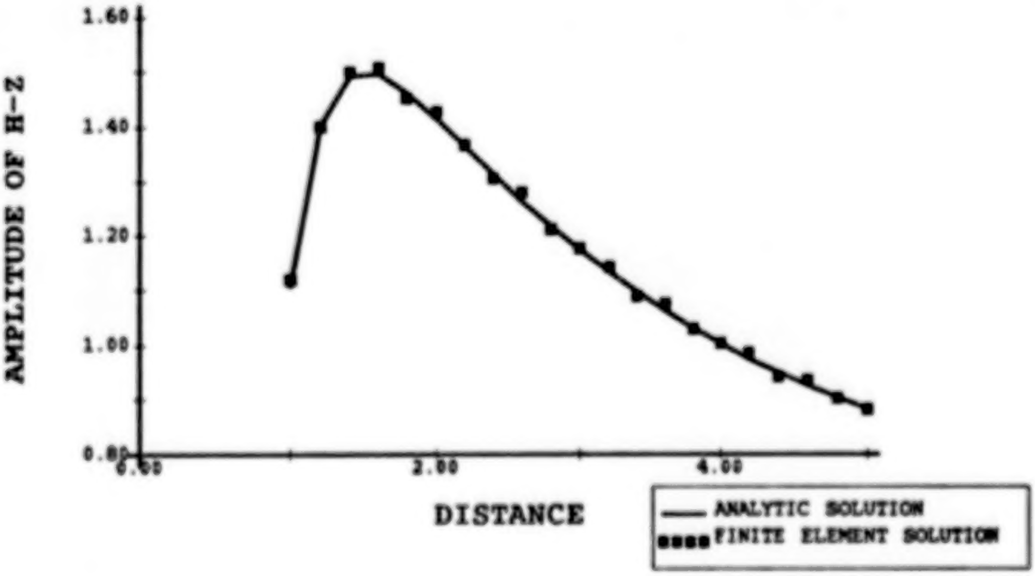


Fig. 34. AMPLITUDE OF THE BACKSCATTERED H-Z WAVE

BEST COPY AVAILABLE

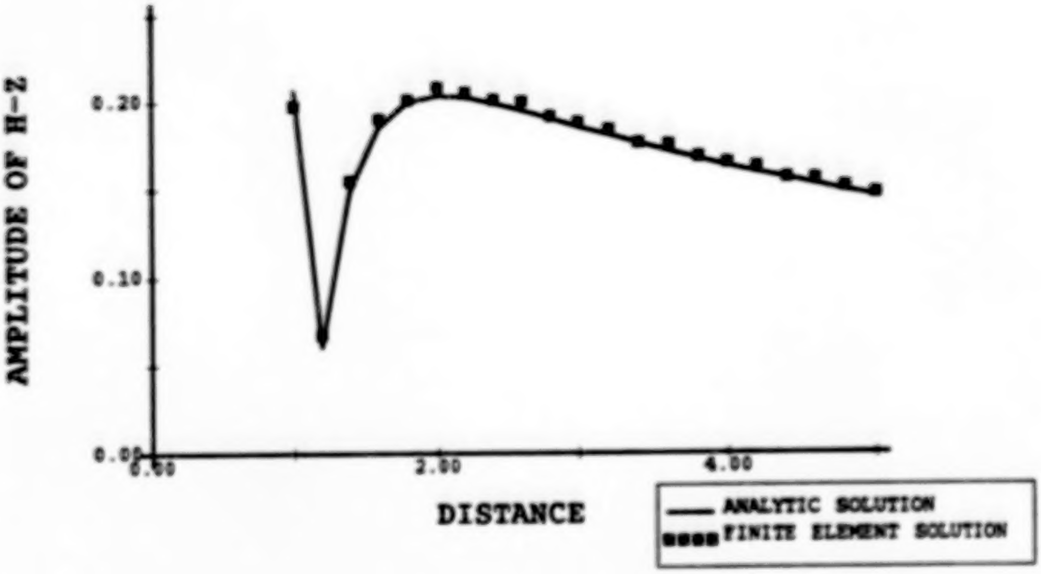


Fig. 35. WORST CASE FOR THE AMPLITUDE OF THE H-Z WAVE

CALCULATION OF LOW FREQUENCY VIBRATIONAL RESONANCES OF SUBMERGED STRUCTURES

by

Gordon C. Everstine

Applied Mathematics Division (184)
David Taylor Research Center
Bethesda, Maryland 20084 U.S.A.

ABSTRACT

Numerical techniques for calculating the low frequency vibrational resonances of submerged structures are reviewed. Both finite element and boundary element approaches for calculating fully-coupled added mass matrices for use in NASTRAN analysis are described and illustrated. The finite element approach is implemented using existing capability in NASTRAN. The boundary element approach uses the NASHUA structural-acoustics program to compute the added mass matrix. The two procedures are compared to each other for the case of a submerged cylindrical shell with flat end closures. It is concluded that both procedures are capable of computing accurate submerged resonances and that the more elegant boundary element procedure is easier to use but may be more expensive computationally.

INTRODUCTION

One problem of interest in numerical structural-acoustics is that of determining the natural vibrational frequencies of general submerged structures. At low frequencies, it is known[1] that the fluid appears to the structure like an added mass (i.e., the fluid pressure on the wet surface is in phase with structural acceleration). At higher frequencies, the fluid impedance (the ratio of fluid pressure to velocity) is mathematically complex, since it involves both mass-like and damping-like effects. The primary difference between these two situations from a computational point of view is that the low frequency calculation can be performed using standard real eigenvalue analysis techniques, whereas the higher frequency calculation requires more expensive complex eigenanalysis. In addition, as frequency increases, the added mass effects decline and the damping (or piston) effects increase, so that the interpretation of the complex eigenvectors as "normal modes" becomes more difficult. For shell structures, such complications become somewhat academic, since shells have high modal density above the first few modes, making the usefulness of computing such modes in doubt anyway.

Consequently, for this paper, we restrict our interest to the calculation of low frequency modes, in which case the finite element calculation of submerged resonances reduces to that of computing the added mass effects of the surrounding fluid on the structure. The added mass calculation requires solving Laplace's equation in the fluid domain exterior to the structure, a calculation which can be performed using either finite element or boundary element techniques, among others. Here we describe the NASTRAN computation of submerged natural frequencies using both approaches. We start by summarizing the relevant theory and then illustrate the two approaches using as an example the vibrations of a submerged cylindrical shell with flat end closures.

THEORETICAL APPROACHES

Consider an arbitrary three-dimensional elastic structure submerged in a heavy fluid like water. The structure is modeled mathematically using the equations of elasticity and the engineering approximations for beams, plates, and shells. A finite element model of a free, undamped structure yields the matrix equation

$$M\ddot{u} + Ku = 0, \tag{1}$$

where M and K are the structural mass and stiffness matrices, respectively, and u is the vector of displacement components. The fluid is modeled mathematically as a medium for which the pressure satisfies (in the time domain) the scalar wave equation[2,3]

$$\nabla^2 p = \ddot{p}/c^2, \tag{2}$$

where c is the speed of sound in the fluid. At the fluid-structure interface, momentum and continuity considerations require that the fluid pressure be applied to the structure and that the normal derivative of pressure be proportional to normal acceleration:

$$\partial p/\partial n = -\rho \ddot{u}_n, \tag{3}$$

where n is the outward normal (from the structure into the fluid) at the interface, and ρ is the mass density of the fluid. We consider two numerical approaches to treating the fluid domain: finite element and boundary element.

Finite Element Approach

Since the scalar wave equation (2) is a special case of the vector wave equation satisfied by the structural displacements, the fluid domain can be modeled using the same types of elastic finite elements used to model the structure if an analogy is drawn between structural displacement and fluid pressure.[4] Thus, if finite elements are used to model both structure and fluid, the system of coupled equations which results is of the form

$$\begin{bmatrix} M & 0 \\ -\rho L^T & Q \end{bmatrix} \begin{Bmatrix} \ddot{u} \\ \ddot{p} \end{Bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix} \begin{Bmatrix} \dot{u} \\ \dot{p} \end{Bmatrix} + \begin{bmatrix} K & L \\ 0 & H \end{bmatrix} \begin{Bmatrix} u \\ p \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}, \tag{4}$$

where p is the vector of fluid pressures at the fluid grid points, Q and H are the fluid counterparts to the structural mass and stiffness matrices, respectively, -L is the rectangular area matrix which converts a vector of fluid pressures (positive in compression) at the wet structural points to a vector of forces at all points in the output coordinate systems selected by the user, and C is a radiation boundary condition matrix with nonzero entries only for fluid DOF on the outer boundary. (Radiation boundary conditions are intended to transmit, rather than reflect, outgoing waves.) A useful alternative to the nonsymmetric system, Eq. 4, is the symmetric potential formulation,[3] which is obtained by transforming from fluid pressure p to fluid velocity potential q (the time integral of pressure) as the fundamental fluid unknown:

$$\begin{bmatrix} M & 0 \\ 0 & Q \end{bmatrix} \begin{Bmatrix} \ddot{u} \\ \ddot{q} \end{Bmatrix} + \begin{bmatrix} 0 & L \\ L^T & C \end{bmatrix} \begin{Bmatrix} \dot{u} \\ \dot{q} \end{Bmatrix} + \begin{bmatrix} K & 0 \\ 0 & H \end{bmatrix} \begin{Bmatrix} u \\ q \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}. \tag{5}$$

To model the fluid with standard elastic finite elements, we let the z-component of displacement represent the velocity potential q, fix all other DOF at fluid grid points, and specify the fluid element elastic properties as[2]

$$G_e = -1/\rho, \quad E_e = -10^{20}/\rho, \quad \nu_e = \text{unspecified}, \tag{6}$$

where the subscript "e" is added to emphasize that these are the values entered on input data cards (e.g., MAT1 in NASTRAN) for the elements. Under the analogy, the element "mass density" ρ_e specified for the fluid is

$$\rho_e = \begin{cases} 0, & \text{incompressible fluid } (c \to \infty) \\ -1/(\rho c^2), & \text{compressible fluid } (c \text{ finite}). \end{cases} \tag{7}$$

This specification of material properties is required for symmetry of the coefficient matrices in Eq. 5.

For large expanses of exterior fluid, only a small portion of fluid need be modeled.[5,6] For an incompressible fluid, the outer boundary may be located at one or two structural diameters away from the structure and a pressure-release (p=0) boundary condition imposed (with SPCs). For a compressible fluid, the outer boundary is located at one or two acoustic wavelengths away from the structure, and dashpots of constant $-A/(\rho c)$ are attached between the fluid DOF and ground to absorb (approximately) the outgoing waves. (This is the plane-wave absorbing boundary condition.)

The above theoretical description allows for the possibility of fluid compressibility effects, which impose requirements on the fluid mesh size and extent and require complex eigenanalysis for the solution of Eq. 4. Since often the interest is in low frequency vibrations, which is equivalent to assuming fluid incompressibility, we specialize the above equations to the case $c \to \infty$. For an incompressible fluid, the matrices Q and C above vanish, and the coupled system (4) simplifies to

$$\begin{bmatrix} M & 0 \\ -\rho L^T & 0 \end{bmatrix} \begin{Bmatrix} \ddot{u} \\ \ddot{p} \end{Bmatrix} + \begin{bmatrix} K & L \\ 0 & H \end{bmatrix} \begin{Bmatrix} u \\ p \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}. \tag{8}$$

An alternative form of Eq. 8 results if the pressure vector p is eliminated from this system to yield

$$(M + M_a)\,\ddot{u} + K\,u = 0, \tag{9}$$

where the symmetric, non-banded matrix $M_a = \rho L H^{-1} L^T$ is referred to as the added mass matrix.

The low frequency (added mass) vibration problem can be solved using the symmetric potential formulation (Eq. 5 with Q and C both zero), the pressure formulation (Eq. 8), or the added mass matrix formulation (Eq. 9). The last form, Eq. 9, has the advantage of being in standard form for a real eigenvalue problem and, moreover, allows the added mass matrix to be calculated using any suitable approach, including boundary elements and finite elements. However, Eq. 9 has the (considerable) disadvantage that matrix bandedness is destroyed, since M_a couples all the wet DOF to each other. If the surrounding fluid domain is modeled with finite elements, the eigenvalue problem can alternatively be solved using Eqs. 5 or 8, which have more DOF than Eq. 9 but remain banded (if the structural and fluid unknowns are properly sequenced). The main distinction between Eqs. 5 and 8 is that the latter involves nonsymmetric coefficient matrices. Although Eq. 8 is a real eigenvalue problem, it can not be solved as such by NASTRAN (because of the nonsymmetry) and must be solved using complex eigenvalue analysis.

Boundary Element Approach

The added mass matrix in Eq. 9 can also be obtained by boundary element techniques.[7-11] In the frequency domain, where the time dependence $\exp(i\omega t)$ is suppressed, the basis for such an approach is the Helmholtz surface integral equation satisfied by the fluid pressure p on the surface S of a submerged structure:

$$\int_S p(\underline{x})(\partial D(r)/\partial n)dS - \int_S q(\underline{x})D(r)dS = p(x')/2, \quad x' \text{ on } S \tag{10}$$

where D is the Green's function

$$D(r) = e^{-ikr}/4\pi r, \tag{11}$$

$$q = \partial p/\partial n = -i\omega\rho v_n, \tag{12}$$

$k = \omega/c$ is the acoustic wave number, c is the speed of sound in the fluid, r is the distance from $\underline{x}$ to $\underline{x}'$ (Fig. 1), ρ is the mass density of the fluid, and v_n is the outward normal component of velocity on S. As shown in Fig. 1, $\underline{x}$ and $\underline{x}'$ in Eq. 4 are the position vectors for points P_j and P_i on the surface S, the vector $\underline{r} = \underline{x}' - \underline{x}$, and $\underline{n}$ is the unit outward normal at P_j. We denote the lengths of the vectors $\underline{x}$, $\underline{x}'$, and $\underline{r}$ by x, x', and r, respectively. The normal derivative of the Green's function D appearing in Eq. 10 can be evaluated as

$$\partial D(r)/\partial n = (e^{-ikr}/4\pi r)(ik + 1/r)\cos\beta, \tag{13}$$

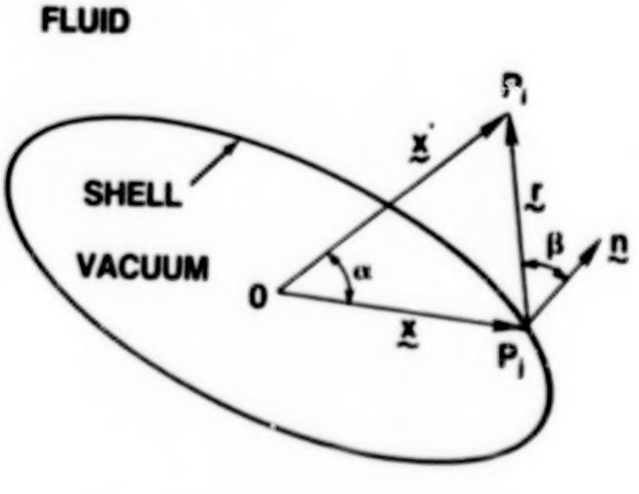


Fig. 1. Notation for Helmholtz integral equation.

where β is defined as the angle between the normal $\underline{n}$ and the vector $\underline{r}$, as shown in Fig. 1.

The substitution of Eqs. 11-13 into the surface equation (10) yields

$$p(\underline{x}')/2 - \int_S p(\underline{x})\ (e^{-ikr}/4\pi r)\ (ik + 1/r)\cos\beta\ dS$$

$$= i\omega\rho \int_S v_n(\underline{x})\ (e^{-ikr}/4\pi r) dS. \qquad (14)$$

This integral equation relates the fluid pressure p and normal velocity v_n on S. If Eq. 14 is discretized for numerical computation (the details of which were presented previously[9]), we obtain the matrix equation

$$Ep = Cv_n \qquad (15)$$

on S. The dimensionality of this system (i.e., the dimension of vectors p and v_n) is f, the number of fluid DOF (the number of wet points on the surface S). Hence, the added mass matrix (the matrix which converts fluid acceleration to fluid force) is, in terms of the fluid DOF,

$$M_n = AE^{-1}C/i\omega, \qquad (16)$$

where A is the diagonal fxf area matrix for the wet surface. As given above, M_n is full, symmetric, frequency-dependent, and complex. The low frequency (incompressible fluid) added mass matrix is obtained by evaluating M_n in the limit $\omega \to 0$. An inspection of the formulas[9] for the entries in the fluid matrices E and C reveals that, for small frequency, E is real and constant, and C is purely imaginary and proportional to ω. Thus, to compute M_n in Eq. 16, we consider only the real parts of E and $C/i\omega$ for small ω. With this interpretation, the added mass matrix M_n is now full, symmetric, real, and independent of frequency.

To relate the f normal DOF on the wet surface to the complete set of s independent structural DOF, we introduce a transformation matrix G, which is defined as the rectangular sxf matrix of direction cosines to transform a vector F_n of outward normal forces at the wet points to a vector F of forces at all points in the output coordinate systems selected by the user. Thus,[10]

$$F = GF_n, \quad v_n = G^T v, \quad \text{and} \quad a_n = G^T a, \qquad (17)$$

where v and a are the velocity and acceleration vectors for the independent structural DOF, respectively, and the subscript n is used to denote the outward normal components of these vectors. For time-harmonic analysis, $v = i\omega u$ and $a = i\omega v$. The transformation matrix G can then be used to transform the added mass matrix displayed in Eq. 17 from normal DOF to the independent structural DOF:

$$M_a = GAE^{-1}(C/i\omega)G^T. \tag{18}$$

Here again, we consider only the real parts of E and $C/i\omega$ for small ω. The matrix M_a given above is the boundary element equivalent of the finite element matrix of the same name defined following Eq. 9. M_a is real, symmetric, non-banded, and independent of frequency. (The symmetry of M_a, while not obvious from the above definition, follows by reciprocity arguments.)

We note that the coupling matrix L defined in Eq. 4 is the product of the transformation and area matrices G and A.

We clearly could have started with the Laplace, rather than the Helmholtz, integral equation and avoided the complex, frequency-dependent matrices.[11] We chose this approach since the four matrices G, A, E, and C needed to compute the added mass matrix M_a are readily available in NASTRAN form from the computer program called SURF, which is part of the NASHUA structural-acoustics package.

NASTRAN IMPLEMENTATIONS

Finite Element Approach

The finite element procedure used here to compute resonances of submerged shells is the symmetric potential formulation as shown in Eq. 5 except that, for incompressible fluids, the matrices Q and C are both zero. To solve this system with NASTRAN,[12] a finite element model is required for both the structure and a portion of the surrounding fluid. The model for the structure is constructed in the usual way. The model for the fluid domain is constructed using any of the general elastic elements which are geometrically compatible with the elements chosen for the structure. Thus, if the structure is modeled with QUAD4s, the fluid should be modeled with IHEX1s; if the structure is modeled with CONEAX elements, the fluid should be modeled with TRIAAX or TRAPAX elements.

Since the z component of displacement represents, by analogy (in both Cartesian and cylindrical coordinate systems), the scalar velocity potential q in Eq. 5, all other DOF at fluid mesh points are eliminated by single point constraints. The material properties are assigned to the fluid elements according to Eqs. 6 and 7a. If the fluid is considered to be of infinite extent, the finite element model of the fluid should be truncated not closer than one shell diameter away from the shell, where a pressure-release (q=0) boundary condition is imposed.

The coupling matrix L is entered as a symmetric "damping" matrix using NASTRAN's direct matrix input (DMIG) data cards. L has nonzero entries only at the intersections of matrix columns associated with interface fluid DOF with rows associated with the translational DOF of coincident structural points. Each nonzero entry of L is a component of the outwardly-directed area vector, which is a normal vector whose magnitude is equal to the area assigned to a wet point. The resulting system is solved using NASTRAN's direct complex eigenvalue analysis (Rigid Format 7) because of the presence of the coupling matrix in the "damping" matrix. (However, since there is no actual damping, all the natural frequencies are real.)

Since both structural and fluid DOF are included in the finite element model, the interpretation of tabular output is aided if only the structural DOF are printed and the printing of the fluid unknowns is suppressed.

Boundary Element Approach

The boundary element generation of the added mass matrix is implemented using the fluid matrix generation capability available in the NASHUA processor called SURF.[9] For each unique set of symmetry constraints, the procedure involves two steps, the first of which is identical to the first step of a NASHUA structural-acoustic analysis. In general, this step is a NASTRAN analysis whose primary purpose is to generate an OUTPUT2 file containing geometric information and the definition of the wet surface of the structure. The second step (described here for the first time for added mass matrix generation) involves the sequential execution of SURF (which generates the matrices G, A, E, and C appearing in Eq. 18) followed by NASTRAN for the real eigenvalue analysis. For completeness, we describe both steps in the boundary element approach to compute submerged resonances.

The first step is a modified NASTRAN direct frequency response analysis in which the structure is defined and an outwardly-directed static unit pressure load applied to the wet faces of all elements in contact with the exterior fluid. This load, which is invoked using the case control card LOAD, is used to generate areas and normals. In addition, the following DMAP Alter is inserted into the Executive Control Deck:

```
ALTER    1 $ NASHUA STEP 1, COSMIC 1988 RF8 (REVISED 12/14/87)
ALTER    21,21 $ REPLACE GP3
GP3      GEOM3,EQEXIN,GEOM2/SLT,GPTT/S,N,NOGRAV/NEVER=1 $ SLT
ALTER    117,117 $ REPLACE FRRD
SSG1     SLT,BGPDT,CSTM,SIL,EST,MPT,GPTT,EDT,MGG,CASECC,DIT/
         PG/LUSET/NSKIP $ PG
SSG2     USET,GM,YS,KFS,GO,DM,PG/QR,PO,PS,PL $ PL
OUTPUT2  BGPDT,EQEXIN,USET,PG,PL $
OUTPUT2  CSTM,ECT,,, $
OUTPUT2  ,,,, //-9 $
EXIT     $
ENDALTER $
```

The UT1 file created by OUTPUT2 must be saved after the NASTRAN execution.

The second step in this procedure consists of the sequential execution of the NASHUA processor SURF followed by NASTRAN. SURF reads the UT1 file generated in Step 1 and generates the matrices G, A, E, and C appearing in Eq. 18. These matrices are written in NASTRAN's INPUTT2 format. Since SURF generate's the frequency-dependent fluid matrices E and C for compressible fluids, a small (but nonzero) frequency must be specified as input for the generation of these matrices. The nondimensional frequency ka = 0.01 is a reasonable choice, where a is a typical length (e.g., radius) of the structure. Following SURF, a modified NASTRAN real eigenvalue (Rigid Format 3) analysis is performed. The following DMAP Alter is included in this run:

```
ALTER     1 $ ADDED MASS MATRIX, COSMIC 1988 RF3 (REVISED 12/22/88)
ALTER     3 $
INPUTT2   /G,A,CT,E,DAT $ READ FLUID MATRICES FROM SURF
ALTER     69 $ BEFORE READ
PARAML    DAT//*DMI*/1/2/FREQ $ GET FREQ FROM DAT
PARAMR    //*COMPLEX*//FREQ/0./FREQC $ FREQ+I*0
PARAMR    //*MPYC*////W/FREQC/(6.283185,0.) $ OMEGA
PARAMR    //*MPC*////IW/W/(0.,1.) $ I*OMEGA
PARAMR    //*DIVC*////IWI/(1.0,0.0)/iw $ 1/IW
DIAGONAL A/PVECF/*COLUMN*/0. $ FLUID SET VECTOR OF 1'S
DIAGONAL KAA/PVECA/*COLUMN*/0. $ A-SET VECTOR OF 1'S
ADD       CT,/CTIW/IWI $ CT/IW
PARTN     CTIW,PVECF,/,,CTIWR,/1/1 $ EXTRACT REAL PART OF CT/IW
PARTN     E,PVECF,/,,ER,/1/1 $ EXTRACT REAL PART OF E
TRNSP     G/GT $
TRNSP     CTIWR/CIW $
SOLVE     ER,CIW/EICIW $
MPYAD     A,EICIW,/AEICIW/1 $ REAL NONSYM ADDED MASS (FLUID DOF)
PARTN     AEICIW,PVECF,/,,MADDF,/1/1///6 $ REAL SYM ADDED MASS (F-DOF)
MPY3      GT,MADDF,/MADDS $ REAL SYM ADDED MASS (STRUCTURAL DOF)
ADD       MAA,MADDS/MSUM $ STRUCTURAL + ADDED MASS
EQUIV     MSUM,MAA $ REPLACE MAA WITH SUM
ENDALTER $
```

This Alter combines the input matrices G, A, E, and C to form the added mass matrix M_a according to Eq. 18 and replaces the structural mass matrix M with the sum $M + M_a$. To assure compatibility with the frequency used in SURF, the frequency is passed from SURF to NASTRAN for use in this Alter. The real parts of the complex matrices E and $C^T/i\omega$ are extracted as soon as possible in the Alter so that less expensive real arithmetic can be performed as much as possible. (Real parts of matrices can be extracted by executing a dummy PARTN module to redefine a complex matrix as real. The same trick can also be used to declare as symmetric a symmetric matrix which NASTRAN thinks is nonsymmetric.)

The use of checkpoint may optionally link the two steps in this procedure. However, in any case, the user must ensure that NASTRAN's internal grid point sequence (as generated by the BANDIT module) is the same in both steps. Otherwise, the sequence used for the generation of the SURF matrices would not agree with that used when the matrices are combined in the second step.

NUMERICAL EXAMPLE

We illustrate these procedures by computing, using both finite element and boundary element techniques, the fluid-loaded resonances of a submerged cylindrical shell with flat end closures. The particular problem solved has the following characteristics:

$a = 5$ m	mean shell radius
$L = 60$ m	shell length
$h = 0.05$ m	shell thickness (shell and end plate)
$E = 1.96 \times 10^{11}$ N/m^2	Young's modulus
$\nu = 0.3$	Poisson's ratio
$\rho_s = 7900$ kg/m^3	shell density
$\rho = 1000$ kg/m^3	fluid density
$c = 1500$ m/sec	fluid speed of sound

For the finite element model of both structure and fluid, a half-length model was prepared using axisymmetric elements (the conical shell CONEAX for the shell and the triangular ring TRIAAX for the fluid), as shown in Fig. 2. The structural model consisted of 25 elements over the half-length and four elements on the end plate. The outer boundary of the fluid model was located about 16 meters from the axis of the shell. Symmetry conditions were imposed at the mid-length, thus restricting the available modes to those symmetric with respect to the mid-length. The NASTRAN bulk data for this model were generated automatically by a special purpose fluid-structure data generator called SFG written by Richard J. Kazden of the David Taylor Research Center.

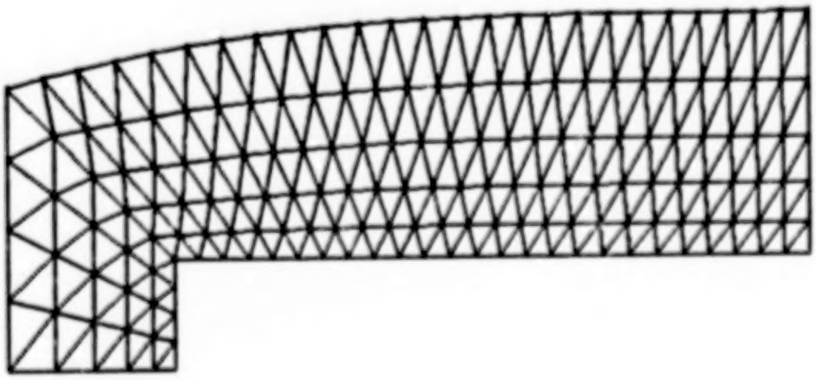


Fig. 2. Axisymmetric finite element model of structure and fluid.

For the analysis with added mass effects generated by boundary elements, a general shell model of the structure was prepared using NASTRAN's four-node isoparametric membrane/bending quadrilateral plate element QUAD4. A quarter model was prepared (half the length and half the circumference) using 25 elements longitudinally, 12 elements circumferentially, and four elements radially on the end plate, as shown in Fig. 3. Symmetry was imposed at both planes of geometric symmetry. Since all fluid effects were computed by the NASHUA processor SURF, no fluid mesh was required.

Four analyses were performed for this problem:

1 - conical shell model, in-vacuo, circumferential harmonic $n \leq 5$, 715 DOF,

2 - QUAD4 model, in-vacuo, 2093 DOF,

3 - conical shell model, fluid-loaded, finite element added mass effects, $n \leq 5$, 1465 DOF, and

4 - QUAD4 model, fluid-loaded, boundary element added mass effects, 2093 DOF (matrices not banded).

The first 21 natural frequencies and mode shapes were found among those which have circumferential index $n \leq 5$ and are symmetric with respect to the mid-length plane. The results of these calculations are shown in the table on the next page. The second column in the table (Harm. n) denotes the circumferential harmonic index, the number of full waves around the circumference. (For the end plate, n thus denotes the number of nodal diameters.) The third column (Shell m) denotes the number of longitudinal half waves. The fourth column (Plate m) denotes the number of nodal circles (plus one) in the end plate. The next two columns of the table list the in-vacuo natural frequencies (in Hz.) of the cylindrical shell for both the conical shell and QUAD4 models. The next two columns of the table list the

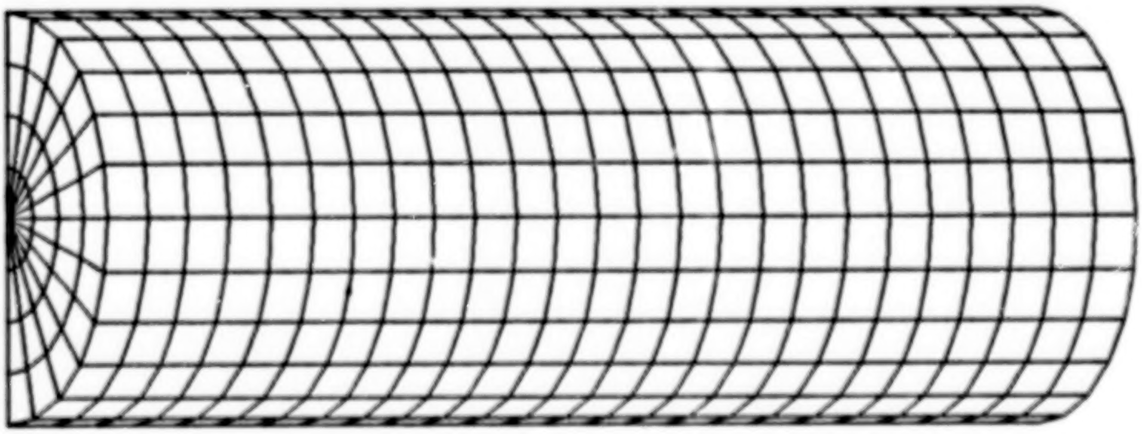


Fig. 3. QUAD4 finite element model of cylindrical shell.

Table. In-Vacuo and Fluid-Loaded Natural Frequencies of Cylindrical Shell with Flat End Plates

	Mode			Frequency (Hz)				
No.	Harm. n	Shell m	Plate m	CONEAX in-vacuo	QUAD4 in-vacuo	CONEAX subm. (F.E. mass)	QUAD4 subm. (B.E. mass)	Approx. Theory
1	1	0		0	0	0	0	0
2	2	1		2.72	2.72	1.13	1.13	1.11
3	3	1		3.84	3.90	1.79	1.81	1.77
4	0		1	4.27	4.22	1.63	1.44	1.38
5	4	1		7.04	7.19	3.61	3.67	3.57
6	4	3		9.29	9.34	4.81	4.82	4.70
7	1		1	9.53	9.20	4.44	4.26	4.22
8	3	3		10.4	10.4	4.94	4.93	4.82
9	5	1		11.3	11.6	6.31	6.38	6.18
10	5	3		12.2	12.4	6.83	6.86	6.67
11	1	3		13.4	13.3	7.04	6.88	
12	2		1	15.6	15.1	8.31	8.02	
13	5	5		15.8	15.9	8.99	8.88	8.65
14	0		2	15.9	16.4	8.66	8.40	
15	4	5		17.0	16.9	8.94	8.85	8.66
16	2	3	3	18.6	18.5	8.05	8.07	7.75
17	3		1	22.7	22.4	13.2	13.0	
18	5	7		22.9	22.8	13.2	12.9	12.6
19	3	5		24.5	24.1	11.9	11.9	11.5
20	1		2	26.6	27.2	15.4	15.5	
21	4	7		28.4	28.2	15.3	15.1	14.7

fluid-loaded (fully submerged) resonances of the shell using both models. The added mass effects were computed for the conical shell and QUAD4 models using, respectively, the finite element and boundary element techniques described above.

The last column of the table lists approximate theoretical predictions for the fluid-loaded resonances. These values are computed in the following way. In general, since frequency is inversely proportional to the square root of mass, the ratio of submerged to in-vacuo resonant frequencies for a structure is

$$f_{wet}/f_{dry} = (1 + M_a/M)^{-1/2}, \qquad (19)$$

where M_a and M are the added mass and structural mass, respectively. For both plates and cylindrical shells, this ratio of added to structural mass can be written in the form[13,14]

$$M_a/M = \alpha(\rho/\rho_s)(a/h), \tag{20}$$

where α is a dimensionless parameter which depends on the boundary conditions, modal wavenumbers, and, for the case of a cylinder, the length-to-radius ratio. For a finite length, simply supported cylindrical shell, α can be approximated for the lobar ($n > 1$) modes as[13]

$$\alpha = n^2/((n^2 + 1)(n^2 + (m\pi a/L)^2)^{1/2}). \tag{21}$$

For a clamped circular plate,[14] $\alpha = 0.6689$ for the (0,1) mode and 0.3087 for the (1,1) mode, where the two mode numbers denote, respectively, the number of nodal diameters and the number of nodal circles plus one. Since the conditions under which these relations apply are approximately satisfied with our example, we include their predictions in the table for reference. The ratio, Eq. 19, is applied in the table to the average of the two in-vacuo predictions.

As indicated in the table, most of these 21 modes are either predominantly shell modes or predominantly end plate modes. For one mode (16), the shell and end plate are both active participants in the modal behavior (although with varying levels of relative participation, depending on the model and whether there was fluid loading).

The results in the table show generally very good agreement between the predictions of the two approaches, both in-vacuo and fluid-loaded, even for circumferential harmonics 4 and 5. For these two harmonics, the QUAD4 mesh has only six and 4.8 elements per wavelength in the circumferential direction, respectively, but still does surprisingly well. The two numerical approaches show agreement to within about 2% for all the fluid-loaded modes which exhibit predominantly shell behavior. The two fluid-loaded predictions for the end plate modes all agree to within about 4%, with the exception of Mode 4, the fundamental drum head mode of the plate, where the difference is about 12%. In view of the similarity of the boundary element prediction to the approximate theoretical prediction, the boundary element result is probably the better of the two numerical predictions, perhaps indicating that the finite element mesh used (Fig. 2) needs to be extended a little farther out at the end of the structure.

DISCUSSION

From the results presented in the preceding section, we conclude that both the finite element and boundary element procedures are capable of computing accurate added mass effects due to fluid loading on fully submerged structures. Of the two approaches, the boundary element procedure is the easier to use, since it is highly automated and does not require the generation of a fluid mesh. Even general purpose automatic mesh generators cannot completely solve the fluid meshing issue, since they cannot generate the fluid-structure interface condition, which requires direct matrix input of surface areas. On the other hand, the finite element procedure is somewhat more general, since it can also treat structures which are near a free surface (or other boundary) or are partially submerged.[5] The boundary element procedure used is applicable only to deeply submerged structures (i.e., structures far enough from a fluid boundary so that the boundary can be ignored).

For structures similar to the cylindrical shell considered here, the finite element procedure is also computationally less expensive than the boundary element procedure. This difference is due primarily to the exploitation by the finite element method of the banded matrices which occur with long, slender structures. Consider, for example, the QUAD4 model of the cylindrical shell shown in Fig. 3. The in-vacuo model had 2093 independent DOF with an average matrix wavefront of 79. When the (boundary element) added mass matrix was combined with the structural mass matrix, the average wavefront increased about five-fold to 394. With the eigensolution time proportional to the product of the order of the matrix and the square of the wavefront, the solution time for the submerged case increases by a factor of about 25. On the other hand, a finite element model of a portion of the surrounding fluid would typically double both the matrix order and the matrix wavefront (compared to the in-vacuo case) since each structural grid point (with six DOF) would require about six fluid grid points (each with one DOF) to be added to the model. Such a model would therefore cost only about eight times as much to run as the "dry" model. The trade-off between the finite element and boundary element procedures for solving the underwater vibration problem thus reduces to a trade-off of engineeting time with computer time.

It is concluded therefore that both the finite element and boundary element procedures are capable of computing the fluid loading effects needed for underwater resonance calculations and that the more elegant boundary element approach is easier to use but may be more expensive computationally.

REFERENCES

1. T.L. Geers, "Residual Potential and Approximate Methods for Three-Dimensional Fluid-Structure Interaction Problems," J. Acoust. Soc. Amer. 49, 1505-1510 (1971).

2. G.C. Everstine, "Structural-Acoustic Finite Element Analysis, with Application to Scattering," in Proc. 6th Invitational Symposium on the Unification of Finite Elements, Finite Differences, and Calculus of Variations, edited by H. Kardestuncer (Univ. of Connecticut, Storrs, Connecticut, 1982), pp. 101-122.

3. G.C. Everstine, "A Symmetric Potential Formulation for Fluid-Structure Interaction," J. Sound and Vibration 79, 157-160 (1981).

4. G.C. Everstine, "Structural Analogies for Scalar Field Problems," Int. J. Num. Meth. in Engrg. 17, 471-476 (1981).

5. M.S. Marcus, "A Finite-Element Method Applied to the Vibration of Submerged Plates," J. Ship Research 22, 94-99 (1978).

6. A.J. Kalinowski and C.W. Nebelung, "Media-Structure Interaction Computations Employing Frequency-Dependent Mesh Sizes with the Finite Element Method," The Shock and Vibration Bulletin 51, Part 1, 173-193 (1981).

7. L.H. Chen and D.G. Schweikert, "Sound Radiation from an Arbitrary Body," J. Acoust. Soc. Amer. 35, 1626-1632 (1963).

8. D.T. Wilton, "Acoustic Radiation and Scattering From Elastic Structures," Int. J. Num. Meth. in Engrg. 13, 123-138 (1978).

9. G.C. Everstine, F.M. Henderson, E.A. Schroeder, and R.R. Lipman, "A General Low Frequency Acoustic Radiation Capability for NASTRAN," in Fourteenth NASTRAN Users' Colloquium (National Aeronautics and Space Administration, Washington, DC, 1986), NASA CP-2419, pp. 293-310.

10. G.C. Everstine, F.M. Henderson, and L.S. Schuetz, "Coupled NASTRAN/Boundary Element Formulation for Acoustic Scattering," in Fifteenth NASTRAN Users' Colloquium (National Aeronautics and Space Administration, Washington, DC, 1987), NASA CP-2481, pp. 250-265.

11. J.A. DeRuntz and T.L. Geers, "Added Mass Computation by the Boundary Element Method," Int. J. Num. Meth. in Engrg. 12, 531-549 (1978).

12. "NASTRAN User's Manual," NASA SP-222(08), Computer Software Management and Information Center (COSMIC), University of Georgia, Athens, Georgia (1986).

13. M.C. Junger and D. Feit, Sound, Structures, and Their Interaction (The MIT Press, Cambridge, Massachusetts, 1986), 2nd ed.

14. R.D. Blevins, Formulas for Natural Frequency and Mode Shape (Van Nostrand Reinhold Company, New York, 1979).

Power Flows and Mechanical Intensities in Structural Finite Element Analysis

by

Stephen A. Hambric
Applied Mathematics Division (184)
David Taylor Research Center
Bethesda, MD 20084-5000

ABSTRACT

The identification of power flow paths in dynamically loaded structures is an important, but currently unavailable, capability for the finite element analyst. For this reason, methods for calculating power flows and mechanical intensities in finite element models are developed here. Formulations for calculating input and output powers, power flows, mechanical intensities, and power dissipations for beam, plate, and solid element types are derived. NASTRAN is used to calculate the required velocity, force, and stress results of an analysis, which a post-processor then uses to calculate power flow quantities. The SDRC I-deas Supertab module is used to view the final results. Test models include a simple truss and a beam-stiffened cantilever plate. Both test cases showed reasonable power flow fields over low to medium frequencies, with accurate power balances. Future work will include testing with more complex models, developing an interactive graphics program to view easily and efficiently the analysis results, applying shape optimization methods to the problem with power flow variables as design constraints, and adding the power flow capability to NASTRAN.

INTRODUCTION

Structure-borne sound is the vibrational energy which travels through dynamically loaded mechanical systems. This vibrational energy is radiated eventually into an acoustic medium as noise. An example cited by Wohlever and Bernhard[1] is an airplane wing loaded by engine vibrations. The vibrational energy travels along the wing to the fuselage and is radiated as sound into the cabin. Architects face the problem of structure-borne sound in hotels and apartment buildings, where vibrational energy flows through walls and floors, and is radiated as sound into other rooms. This problem is addressed by

Luzzato and Ortola.[2]

Dynamic analyses are performed to solve these problems, which output exorbitant amounts of data. The analyst is then faced with the problem of interpreting the output. Tabular printouts can be analyzed, spectrum plots generated, and deformed shapes plotted, all of which are useful methods of defining the state of a structure. Another way of quantifying the propagation of structure-borne sound is the calculation of power flows. This method will identify the magnitude and direction of the power at any location in a structure, helping an analyst to find the dominant paths of energy flow and the energy sinks for a given problem. The understanding of the paths of energy which flow from a vibration source (such as the engine in the aircraft example) to certain parts of a structure (the cabin for example) would help an engineer to more easily pinpoint and correct vibration problems.

The important terms used in this study are: power flow, which is actually power, or energy flow, but is termed power flow by the scientific community of this field; mechanical intensity, which is power flow per unit area; and power dissipation, which is the time rate of energy dissipated in a structure. Four main methods for identifying dominant power sources and power flow paths are addressed in the literature: experimental methods, statistical energy analysis, the finite element method, and the power flow method.

Experimental solutions are the most common in the literature. The authors of some of these papers[3–7] use multiple transducers and digital signal processing techniques to solve various power flow problems. A common method is the calculation of cross spectral densities, where two accelerometers are placed a known distance apart on a structure, and response spectra are generated for the two measurement locations. The correlation between the spectra is statistically analyzed, and power flows are computed over some range of frequencies. This approach is similar to the two-microphone technique used by acousticians to solve noise propagation problems in fluid media. Once an experimental apparatus is set up, the analyst may easily vary applied loads and loading frequencies. Unfortunately, accuracy problems may occur due to the added weights and inertias of the transducers attached to an experimental structure.

Statistical energy analysis (SEA) is a computational method used to solve energy flow problems in the high frequency domain. A definitive reference on SEA, although now out of print, is the text by R.L. Lyon.[8] A brief summary of SEA follows. Large structures are split into smaller subsystems; a modal density is estimated for each subsystem so the number of modes in a given frequency band can be determined; dissipation loss factors, which relate energy stored to power dissipated, are estimated for the subsystems; and coupling loss factors, which relate differences of modal energy of subsystems to power flow, are assigned to the junctions of the subsystems. The energy distribution, power flows, and power dissipations are then computed.

SEA is a reasonable way of solving for the average response of structures at high frequencies; however all spatial variations of the power flow field in the substructures remain unknown. A more discrete method must be used to identify specific power flow paths through a structure. Finite element analysis (FEA) may be used for this purpose, but is only cost effective for low to mid-range frequencies, since higher mode shapes are more complex, wavelengths are shorter, and denser finite element meshes are required to model a problem correctly. Mickol and Bernhard[9] succesfully used FEA to identify power flow paths in simple beam and plate structures excited at low frequencies.

Recently, some scientists have proposed a new method to solve for power flows in the middle frequency range. Cuschieri,[10] Nefske and Sung,[11] and Wohlever and Bernhard[1] have been studying this new approach: a finite element analogy where input power is substituted for input force and the power flow field may be solved for directly using a finite element solution.

In this paper FEA is used to solve the power flow problem. For lower modes the method is accurate, models are simple to build and modify using modern modeling software, and analysis results are viewed easily using post-processing graphics packages. Since no commercial software contains a power flow capability, the formulations and computer methods are developed here for NASTRAN.[12] The FEA studies presented in the literature consider only the contribution of flexural wave motion to power flow. Other motion types of power flow, such as axial and torsional, are ignored. In this study, all types of power flow are considered.

First, general methods and formulations for power flows and mechanical intensities, power dissipations, input powers, and output powers are developed for global models, beam elements, plate elements, solid elements, and scalar elements. The required NASTRAN solution, the algorithm of the power flow processor, and the use of I-deas Supertab are outlined. Two test models are analyzed to verify the methods: a simple truss and a beam-stiffened cantilever plate. Finally, based on the results of the test case analyses, conclusions about the method are formed and some thoughts about future directions for work are discussed.

THE FINITE ELEMENT SOLUTION

General Methods

A typical power flow cycle is shown in Fig. 1. The figure shows an arbitrary structure mounted to a connecting structure by a spring and damper coupling. A dynamic load is applied, and energy flows into the structure at the load point. The input power then flows through the structure along multiple

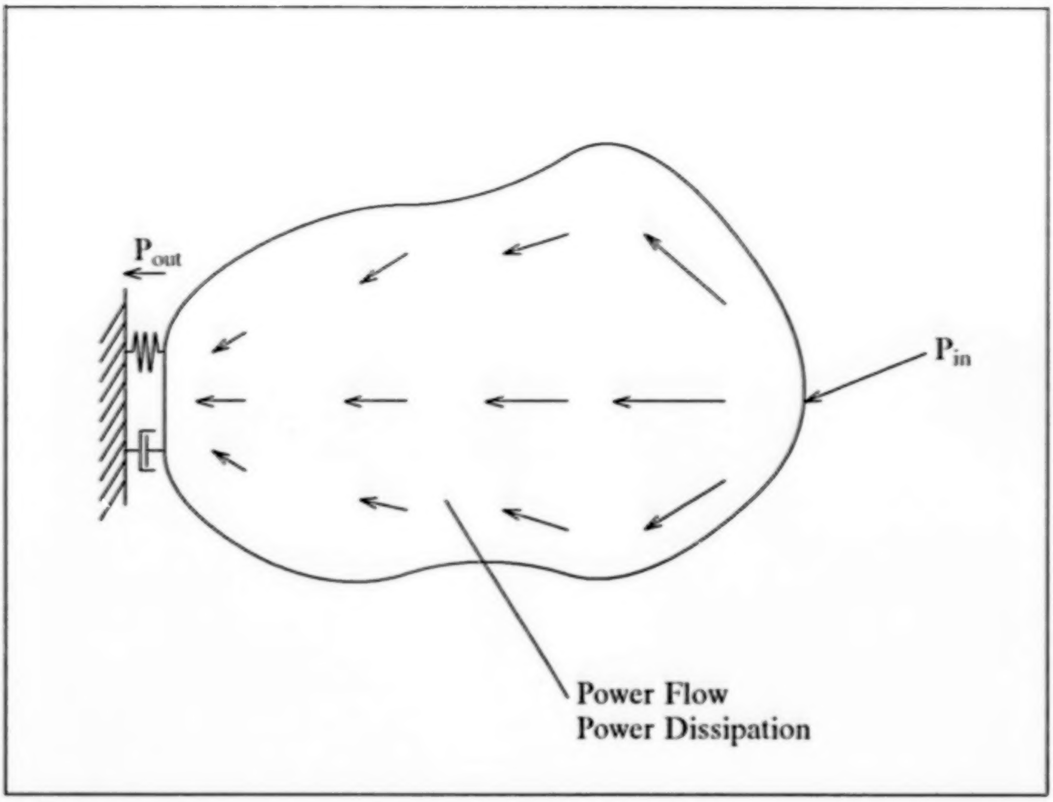


Fig. 1. Sample Power Flow Diagram.

flow paths denoted by arrows, whose lengths represent power flow magnitudes. As the energy flows toward the mounting, it is dissipated by material damping and sound radiation into a surrounding medium, and the flow arrows shorten. The flow and dissipation processes continue until the remaining energy exits the structure through the mounting and flows into the connecting structure. Though only one power entry and exit point is shown in this drawing, multiple loads and mountings may exist. A classic text which describes the flow of structure-borne sound is the book by Cremer, Heckl, and Ungar.[13]

The power flow problem may be solved using NASTRAN. The structure may be modeled using various element types; mountings are modeled using scalar spring, damping, and mass elements; and constraints and loads are directly applied. The steady-state response for the model is solved for a given excitation frequency, and the power flow variables are calculated.

Power Flow and Mechanical Intensity

To calculate power input, power flow, or power output at some location in a given direction, the force in that direction is multiplied by the in-phase part of the velocity in that direction. For example, a bending moment about the x direction is multiplied by the in-phase part of the angular velocity about the x direction. The power flow at that degree of freedom is the real part of that result. This calculation may be visualized as taking the dot product of the force and velocity phasors to solve for the real part of power.

Multiplying one complex number by the in-phase part of another complex number is the same operation as multiplying the first number by the complex conjugate of the other number. Therefore a general formula for power flow in a structure is

$$\text{Power} = Fv^{*}. \qquad (1)$$

Power flow is a complex number. The real part of the calculation is called the active power, and the imaginary part is called the reactive power. The active power is the quantity of interest here.

Mechanical intensity is power flow per unit area, or the stress multiplied by the complex conjugate of velocity. Mechanical intensity is similar to acoustic intensity, which is the pressure in a fluid medium multiplied by the complex conjugate of velocity.

Damping and Power Dissipation

Power may be dissipated in different ways: by material damping, by mountings and surrounding structures, and by radiation as sound. This section discusses the power dissipation due to damping. At this time only material damping is considered in the dissipation process. The effects of sound radiation will be considered in the future.

Power dissipation is calculated differently from power flow and power input. Since power dissipation is the rate of energy dissipation, the energy level of a given element is calculated and multiplied by its damping coefficient. Multiplying the energy dissipation by the angular frequency of excitation gives the power dissipated in that element.

The effects of the material damping coefficient are significant. As the damping coefficient is increased, the power dissipated will increase. If the damping coefficient is zero, no scalar damping elements are applied to the structure, and no sound radiation is considered, power dissipation will be zero and no power flow will exist. This is because, with no damping, forces and velocities will be exactly 90 degrees out of phase, and the in-phase part of velocity is zero. Though this is a physically unrealistic situation, it is one that may occur in a finite element analysis.

To solve for power dissipation, energy dissipation must first be calculated. The energy level in an element is the sum of the element's kinetic energy and potential energy. Since this is a steady-state problem, and the energy is a time-averaged quantity, it may be calculated as twice the kinetic energy:

$$E = mvv^*, \tag{2}$$

where

E = energy,
m = element mass, and
v = velocity.

Power dissipation is then calculated as

$$P_{diss} = 2\pi f \eta E, \tag{3}$$

where

f = rotational frequency, and
η = material damping coefficient.

The ηE term is the energy dissipation, and multiplying by the angular frequency gives the energy dissipation per unit time, or power dissipation. The result will be a real number, since the energy calculation multiplies velocity by its complex conjugate.

The calculation of power dissipation includes the element mass, so the calculation is mesh-dependent. As mesh density increases, element power dissipations decrease. For example, if a beam element were subdivided into two beam elements, the original power dissipation would be split between the two new beams. A way to make the power dissipation calculations mesh-independent would be to divide the results by their respective element masses. At this point, however, the actual power dissipations are calculated because of their importance in checking power balances (see the *Power Balance* section below).

Also, power dissipation is directly related to the mode shapes of an analysis, so areas of large displacements and velocities will be large energy sinks, and nodes (points of near zero displacement) will dissipate almost no power.

Power Input

Power inputs are calculated by multiplying input forces by the complex conjugates of their corresponding velocities. Total input power is calculated as

$$P_{in} = \text{Real}\,[\sum_{i=1}^{n} F_i v_i^*], \tag{4}$$

where

i = load point, and
n = number of loads.

This is a global calculation which is independent of element type.

At this time only force inputs are considered. Other load types may input power to a structure, such as displacements, velocities, and accelerations. The calculation of input powers for these load types will be derived in a subsequent paper.

Power Output

Power output is the power that leaves the system through its mountings and enters the connecting system(s). The external system is modeled using spring, damper, and mass elements. These scalar elements must be connected to additional grid points which are grounded. The forces of constraint are combined with the velocities of the grid attached to the scalar element to calculate power output. The power output is calculated as

$$P_{out} = \text{Real}\,[\sum_{i=1}^{n} F_i v_i^*], \tag{5}$$

where

i = grounded grid, and
n = number of grounded grids.

Power Balance

The terms described above (power input, power dissipation, and power output) are all used to verify a power balance for a given problem. The power balance equation is

$$P_{in} = \sum_{i=1}^{\#elem} P_{diss} + P_{out.} \tag{6}$$

This is the same equation used in SEA theory. Since P_{in}, P_{out}, and P_{diss} are all calculated independently, if the power balance equation holds, then the power flow solution is correct (assuming the original finite element solution is accurate). This power equilibrium equation is therefore an important check on the power formulations and calculations.

Element Formulations

Beam Elements

Most of the literature in the field of power flow is devoted to beams. The landmark paper by Noiseux[14] described methods of measuring the flexural power flow in beams. Many other authors, such as Verheij,[15] Li,[16] Wohlever and Bernhard,[1] and Nefske and Sung[11] have developed power flow capabilities for beam elements.

All the methods developed, however, consider only flexural power flow. Though flexure is arguably the dominant response in a beam, cases arise where axial and torsional response are important. For this reason, all possible components of power flow will be considered.

Power flow methods for BAR element types are derived below. Either lumped mass or coupled mass solutions may be used. Unfortunately, torsional inertia for the BAR element type is not calculated by NASTRAN. Concentrated mass elements with beam torsional inertias entered as masses must be added to the model at the appropriate degrees of freedom (DOF) to solve for accurate torsional power flows.

Power Flow and Mechanical Intensity. A diagram of the BAR element and its force output conventions is shown in Fig. 2.

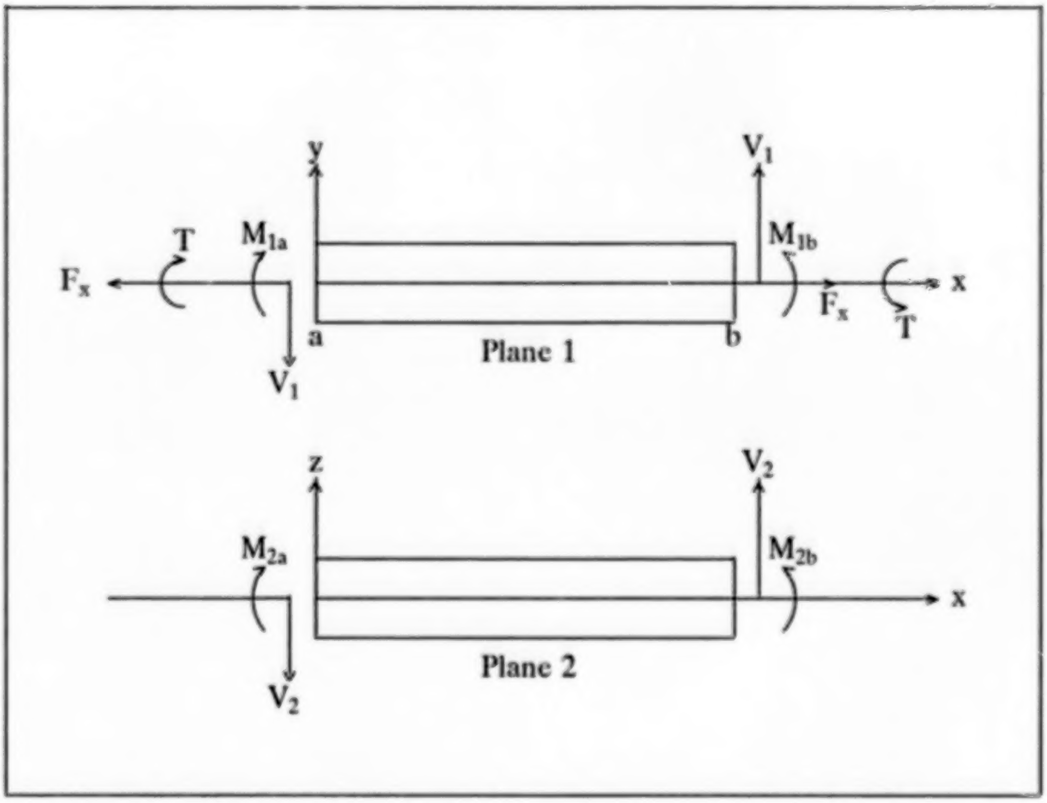


Fig. 2. The BAR Element

Since a beam is a one-dimensional element, energy flows in only one direction: in the local x direction, or along the length of the beam. The total power flow for a beam element is

$$P_x = \text{Real}\left[\, -(F_x v_x^* + V_1 v_y^* + V_2 v_z^* + T\omega_x^* - M_2\omega_y^* + M_1\omega_z^*)\right], \tag{7}$$

where

F_x = axial force,
V_1 = shear force in y direction,
V_2 = shear force in z direction,

T = torsion about x,
M_2 = bending moment about y,
M_1 = bending moment about z,
v_i = translational velocities in direction i, and
ω_i^* = rotational velocities about axis i.

The negative sign in front of the result is due to force and displacement direction conventions for the element. Negative signs appear in the formulations for the plate and solid elements for the same reason. The negative sign in front of the M_2 term is due to the NASTRAN force output convention. In Fig. 2, M_2 is shown as positive in the opposite sense to ω_y. Therefore, $M_2\omega_y^*$ is opposite in sign to the other power flow components.

Velocities are calculated by NASTRAN for each grid point, and beam forces are calculated on an element level. This difference creates a problem, because some way of solving for a power flow on an element level is required. The solution is to solve for a power flow at each grid point, and calculate the average quantity for an element.

Shear, axial, and torsional forces are constant through the element, and are the same for each grid point. Bending moments are calculated at each end of the beam element. Velocities are solved for at a global level, and a coordinate system transformation must be performed to find the velocities at the element level. Since the grid coordinates and beam orientation vector are given, the velocity transformation is straightforward. After power flows at each element end are calculated, they are averaged to give an element power flow. Power flows are then transformed back to NASTRAN's basic coordinate system.

Mechanical intensity is power flow per unit area, and since all the power flow in a beam is along the local x axis, intensity is simply the total power flow divided by the beam's cross sectional area.

Two important observations may be made about power flow in beam elements. Since power flow is one-dimensional in beams, it is independent of mesh variations. Increasing mesh density or varying the mesh pattern will not affect greatly the power flow results (assuming the mesh is dense enough to model accurately the mode shapes of the solution). Also, since power flow is dependent on element force quantities that are discontinuous across element boundaries (axial and shear forces, torsion), the power flow and mechanical intensity quantities are not continuous across beam element boundaries.

Power Dissipation. The energy of a beam element includes both translational and rotatory terms, and is calculated as

$$E = m(v_x v_x^* + v_y v_y^* + v_z v_z^*) + I_{xx}\omega_x\omega_x^* + I_{yy}\omega_y\omega_y^* + I_{zz}\omega_z\omega_z^*, \tag{8}$$

where

m = element mass,

I_{yy}, I_{zz} = mass moments of inertia about cross section,
I_{xx} = polar mass moment of inertia about beam axis,
v_i = local translational velocities in direction i, and
ω_i = local rotational velocities about axis i.

For a lumped mass formulation, the energy terms are calculated at the element centroid; for a coupled mass formulation, they are calculated at the beam ends and averaged. The element energy is then multiplied by $2\pi\eta f$, as in Eq. 3, to yield power dissipation.

The rotational inertial energies are generally small. However, if the beam lengths are long with respect to the cross section, the mass moments of inertia become important. In the case of torsion, where the only large displacement is rotation about the beam's axis, the polar mass moment of inertia term dominates the energy calculation.

Plate Elements

Since the beam element formulation included all components of power flow, power flow capabilities for QUAD elements (QUAD2 and QUAD4), which consider both flexural and membrane effects, are developed.

The literature concerning plate elements is growing, and publications by Mickol and Bernhard,[1] Williams et al,[17] Koshiroi and Tateishi,[18] Noiseux,[13] Fahy and Pierri,[19] and Cuschieri,[20] investigate mechanical intensities and power flows through plate structures. Similar to the literature for beams however, most approaches consider only flexural effects.

Power Flow and Mechanical Intensity. A diagram of a QUAD element and its force and stress output conventions is shown in Fig. 3. The quadrilateral element is two-dimensional, and power may flow in the local x and y directions. The power flow in the x direction is calculated as

$$P_x = \text{Real}\,[\,-(V_x v_z^* - M_x \omega_y^* + M_{xy} \omega_x^* + F_x v_x^* + F_{xy} v_y^*)]; \tag{9}$$

the power flow in the y direction is

$$P_y = \text{Real}\,[\,-(V_y v_z^* + M_y \omega_x^* - M_{xy} \omega_y^* + F_y v_y^* + F_{yx} v_x^*)],$$

where

V_x, V_y = transverse shear forces,
M_x, M_y = bending moments,
M_{xy} = twisting moment,
F_x, F_y = membrane forces,
F_{xy}, F_{yx} = membrane shear,
v_i = local translational velocities in direction i, and
ω_i = local rotational velocities about axis i.

The negative signs in front of the $M_x \omega_y^*$ and $M_{xy} \omega_y^*$ terms are due to the

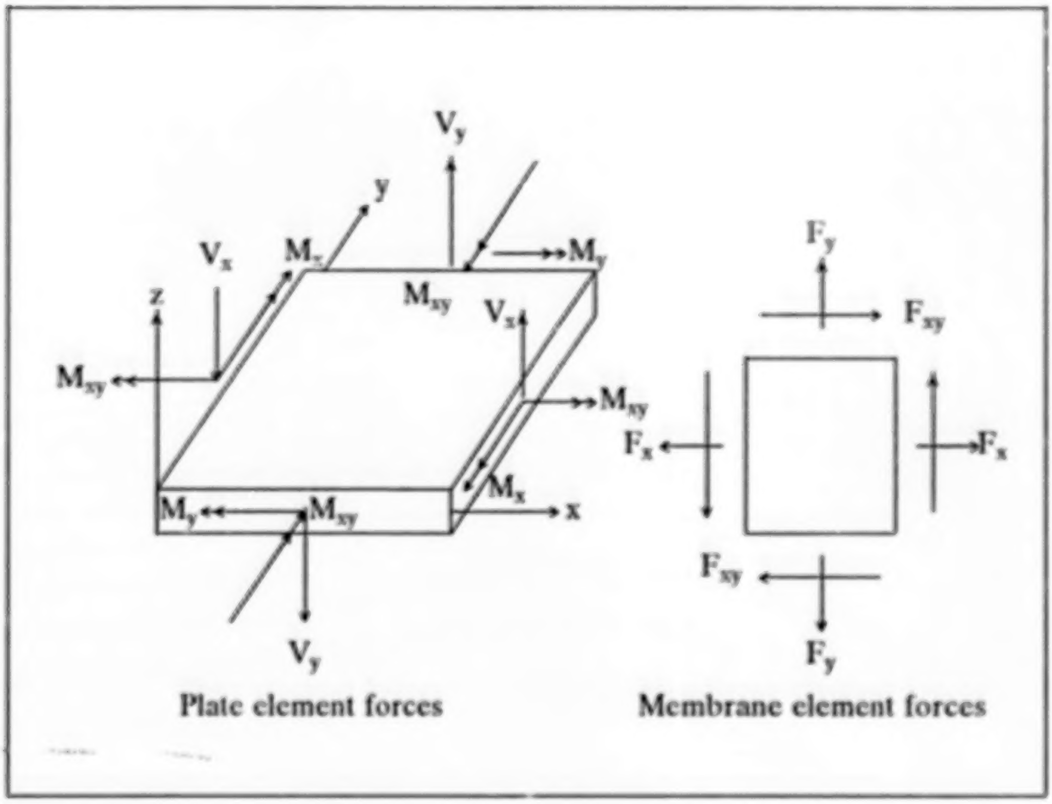


Plate element forces

Membrane element forces

Fig. 3. The QUAD Element

NASTRAN force output convention. These bending moments are opposite in sense to their corresponding rotational velocities.

As in the case of the beam elements, grid velocities must be transformed to the local element coordinate systems to be used in the power flow calculations. After the calculations, the power flow vectors are transformed back to NASTRAN's basic coordinate system.

The above formulation should work for the QUAD2 and QUAD4 element types. Unfortunately, the QUAD4 has not been fully implemented for complex analysis yet, so only the QUAD2 may be used. This is unfortunate, since the QUAD2 is not an isoparametric element, and its membrane performance is poor. In fact, NASTRAN does not calculate membrane forces for the QUAD2, so they must be deduced from the membrane stress outputs.

Calculating membrane forces involves approximating the element's local dx and dy lengths, which combined with the plate thickness will give dA values in the local x and y directions. Multiplying these "side areas" by the stresses will give approximations for the membrane forces. All this additional calculation introduces more error into a membrane formulation which is already poor. If an element is greatly distorted, the membrane results could be completely incorrect.

Finite element meshes for the QUAD2 must be therefore as uniform as possible, since the element is not isoparametric. Also, if membrane effects are dominant in an analysis, the results will be suspect. When the NASTRAN implementation of the QUAD4 is complete, the QUAD4 element will be used.

The mesh dependence of power flow for QUAD elements has not yet been determined.

To calculate mechanical intensities, P_x and P_y are divided by the estimated side areas.

Power Dissipation. The energy of a QUAD element, considering only scalar mass terms, is

$$E = m(v_x v_x^* + v_y v_y^* + v_z v_z^*), \tag{10}$$

where

m = element mass, and
v_i = local translational velocities in direction i.

Element energies are multiplied by ηf to calculate power dissipations. Power dissipation terms are calculated at each grid point and averaged to solve for the element dissipation. This calculation is mesh-dependent, since power dissipation is directly related to element mass.

Solid Elements

Since literature in the power flow field is largely from the experimental sector, solid elements are generally not considered. An experimentalist cannot place a measuring device inside the material of a solid structure. The paper by Pavic,[7] however, describes a method for measuring structural surface intensity. His method, which involves placing transducers on various surfaces of a machinery system to measure two-dimensional mechanical intensities, may be extended to three dimensions. Since a finite element code has no restrictions on making "measurements" internal to a structure, mechanical intensities may be calculated throughout a solid model.

Power Flow and Mechanical Intensity. For the BAR and QUAD elements, force output is given by NASTRAN. For solid elements, stress output is given at grid points and at the element centroid. Pavic[7] uses stresses and velocities in his formulation of structural surface intensities. His formulas for mechanical intensities, extended to three dimensions, are

$$\begin{aligned} I_x &= \text{Real}\,[\,-(\sigma_x v_x + \tau_{xy} v_y + \tau_{xz} v_z)], \\ I_y &= \text{Real}\,[\,-(\sigma_y v_y + \tau_{yx} v_x + \tau_{yz} v_z)], \\ I_z &= \text{Real}\,[\,-(\sigma_z v_z + \tau_{zx} v_x + \tau_{zy} v_y)], \end{aligned} \tag{11}$$

where

I_x, I_y, I_z = global mechanical intensities,
$\sigma_x, \sigma_y, \sigma_z$ = normal stresses,
$\tau_{xy}, \tau_{yz}, \tau_{xz}$ = shear stresses, and
v_i = global translational velocities in direction i.

Since element stresses are given in the basic coordinate system, the velocities do not have to be transformed to element coordinate systems as they were for the BAR and QUAD element types. Calculations are made at each grid point and averaged to calculate the element mechanical intensity.

At this point, no attempt is made to compute power flows using the mechanical intensity results. Although the intensity vector is defined, the problem of finding element face areas in the x, y, and z directions remains.

This formulation is valid for any solid element in NASTRAN, including the linear, parabolic, and cubic isoparametric solid elements. Unfortunately, the complex stress output for each of these element types is incorrect. The stress results in the OESC1 data block are wrong, and when they are passed to the output file processor (OFP) module, errors result in the output file. These errors appear to be related to data types, since asterisks appear in the grid point field of the NASTRAN output file.

The errors associated with the isoparametric elements restricts the usable element types to the HEXA2, which is a superposition of ten tetrahedron elements. As was the case for the QUAD elements, when the NASTRAN errors are fixed, the higher level elements will be used.

The mesh dependence of mechanical intensity for solid elements has not yet been determined.

Power Dissipation. The power dissipation calculations are the same as those for the QUAD elements (Eq. 10). Only mass and translational velocity terms are considered. Power dissipation terms are calculated at each grid point and averaged. Again, since the element mass is directly related to the element energy, power dissipation is mesh-dependent.

Scalar Elements

Scalar elements may be used to simulate mountings and structures connected to the finite element model. ELASi and MASSi elements may be used to model stiffness, damping, and mass effects. These elements may be important for certain analyses, such as when a structure is not rigidly mounted. In certain cases, power may flow out of a structure into an isolator, which will absorb much of the energy, or into a surrounding medium. The accurate modeling of boundary conditions must therefore include scalar element types.

Power Flow, Intensity, and Dissipation. Power flows and power dissipations are not measured in scalar elements, but the presence of external stiffnesses, dampers, and masses may significantly affect the results in the structural element types.

Computer Methods

A flow chart of the solution process is shown in Fig. 4.

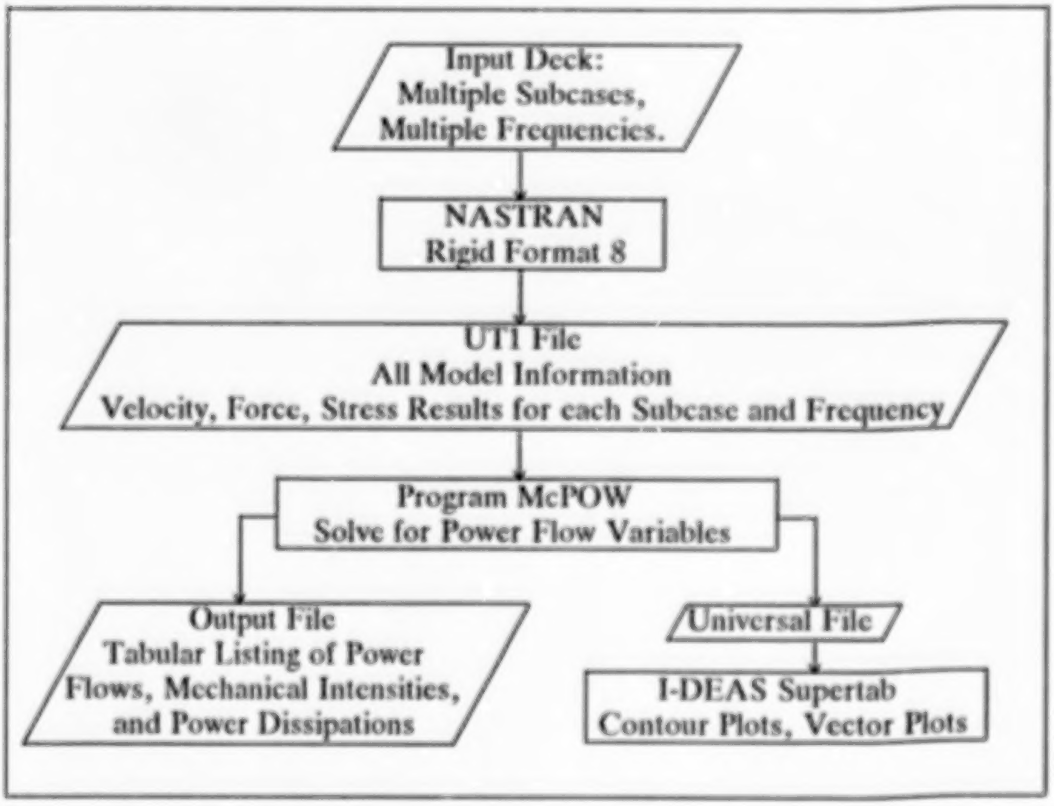

Input Deck:
Multiple Subcases,
Multiple Frequencies.
NASTRAN
Rigid Format 8
UT1 File
All Model Information
Velocity, Force, Stress Results for each Subcase and Frequency
Program McPOW
Solve for Power Flow Variables
Output File
Tabular Listing of Power
Flows, Mechanical Intensities,
and Power Dissipations
Universal File
I-DEAS Supertab
Contour Plots, Vector Plots

Fig. 4. Power Flow Solution Process

NASTRAN's Rigid Format 8 (Direct Frequency Response) is used to solve a given problem for any combination of load cases and excitation frequencies. The model information and problem solution output are written to a UT1 file, which is used as input to the McPOW (Mechanical POWer) program. After the power computations, power flows, mechanical intensities, and power dissipations are written to two output files. One file contains a tabular listing of the power flow results; the other file is formatted as input to the I-DEAS Supertab^{21} post-processor, which is used to interpret visually the results.

NASTRAN Solution

Before running Rigid Format 8, an eigenvalue extraction (Rigid Format 3) can be performed on the model to determine the resonant frequencies and their corresponding mode shapes. Power flows can then be measured at response peaks, and the dominant type of power flow, such as flexural or axial, can be predicted by examining the mode shapes.

Several data blocks must be written to the UT1 output file for the McPOW program. The following ALTER statements are put in the Executive Control Deck:

```
$
$ THE FOLLOWING STATEMENTS CORRESPOND TO THE 1987,88
$ VERSIONS OF COSMIC NASTRAN, RF8
$
ALTER 23$                                AFTER THE TA1 MODULE
OUTPUT2 CASECC,EST,MPT,EQEXIN$
ALTER 135$                               AFTER THE SDR2 MODULE
OUTPUT2 OPPC1,OESC1,OEFC1,OUPVC1$
ENDALTER$
```

In the above alter, the CASECC data block contains case control information, the EST data block holds element information, the MPT data block contains material properties, and the EQEXIN data block holds grid and SIL (Scalar Index List) information. The OPPC1 data block contains the applied forces, the OESC1 data block lists the element stresses, the OEFC1 data block holds element forces, and the OUPVC1 data block contains grid point velocities.

To ensure that all the required data are in the data blocks, the following output requests must be made in the case control deck:

```
FORCE(PHASE)=ALL
STRESS(PHASE)=ALL
VELOCITY(PHASE)=ALL
OLOAD(PHASE)=ALL
```

The capability to calculate power flows for sets of elements will be implemented later.

Power Flow Algorithm

The program McPOW is composed of four main sections: the model information section, the NASTRAN output section, the power flow calculation section, and the output section.

The model information section simply reads the CASECC, EST, MPT, and EQEXIN data blocks from the UT1 file. The NASTRAN output section reads the OPPC1, OESC1, OEFC1, and OUPVC1 data blocks and assigns forces and stresses to element variables, velocities to grid points, and input loads to grid points.

The power flow calculation section first calculates input powers using the input loads and corresponding grid velocities. Next, grid velocities are assigned to elements. Power flows, mechanical intensities, and power dissipations are then calculated using element forces, stresses, and the velocities of the element grids.

The output section writes power flow information to two files. The first contains a tabular listing of the solution variables for each subcase and frequency; the second is a data file in I-DEAS Universal file format.

Post-Processing

The user may analyze the power flow output in two ways: by inspecting the listed output, or using I-DEAS Supertab's post-processor to draw contour plots and arrow plots. Analyzing the tabular output is a good way to check power balances. Power input is equal to power dissipated plus power output. However, to visualize the entire power flow solution in any reasonably complex geometry, a good graphics post-processor is required.

Color contour plots can be used to display power flow magnitudes and power dissipations. Power flow, however, is a vector, and arrow plots are needed to display the direction of the flow. Other authors, such as Heckl,[22] and Koshiroi and Tateishi,[18] have used arrow plots to show power flows in plate structures. An alternative unavailable in I-DEAS Supertab is a combination of a contour and arrow plot, which would illustrate magnitude and direction.

TEST CASES

The test problems illustrate the use of beam, plate, and scalar elements. The QUAD4 and solid elements have not been tested yet.

Simple Truss

Problem Statement

A diagram of a simple truss is shown in Fig. 5. The truss members are constructed of three different types of cross sections. The model was attached to ground at its top and bottom by springs and dampers in all six DOF. The scalar elements simulated the effects of fasteners and the surrounding structure(s). An end load was applied in all six DOF over a range of frequencies. The properties of the two W type sections are given in civil engineering handbooks.

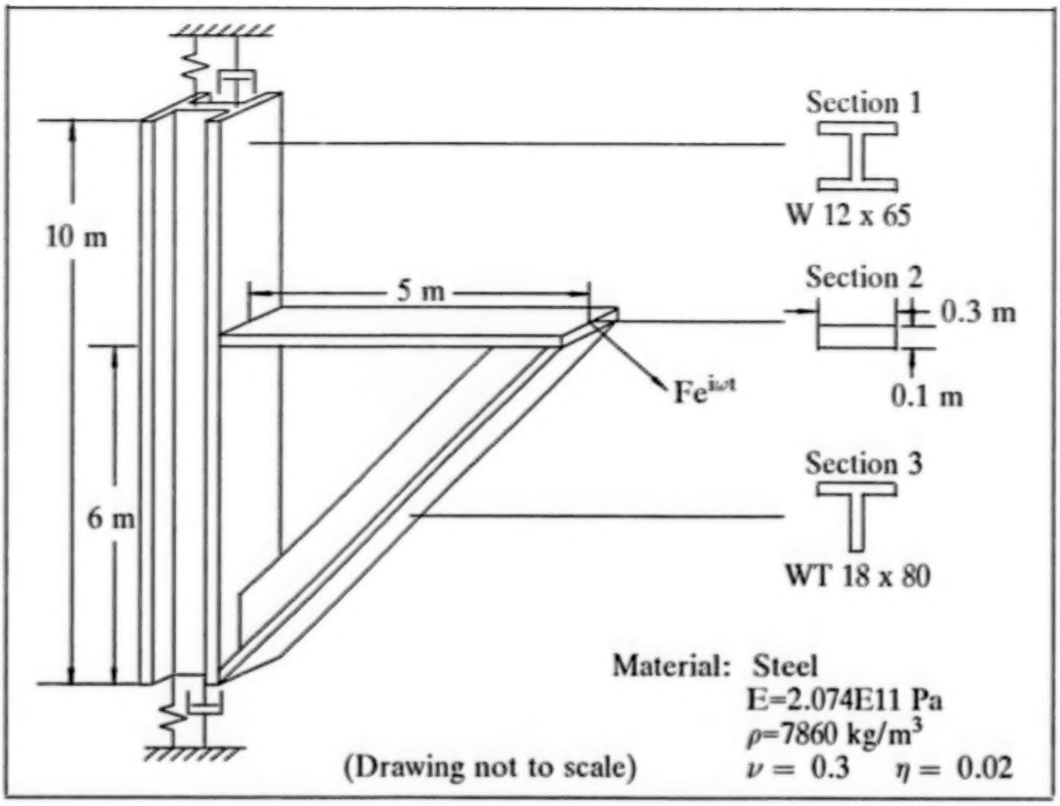

Section 1
W 12 x 65
10 m
5 m
Section 2
0.3 m
$Fe^{i\omega t}$
0.1 m
Section 3
6 m
WT 18 x 80
Material: Steel
E=2.074E11 Pa
ρ=7860 kg/m^3
ν = 0.3 η = 0.02
(Drawing not to scale)

Fig. 5. Simple Truss Problem

The finite element model consisted of 74 BAR elements, with each beam section having a different mesh density. Section 1 was modeled with 40 elements of 0.25 m length, Section 2 consisted of 10 elements of 0.5 m length, and Section 3 was made up of 24 elements of about 0.325 m length. For the scalar elements, spring constants were set at about 100 to 1000 times the stiffness of the members at the appropriate DOF; and the damping constant was set at ten times the material damping constant, or 0.2. This model is a good general test of the power flow methods outlined above, since it has a varying mesh density and the three sections have different beam properties.

Results

The first analysis performed on the model was an eigenvalue extraction (Rigid Format 3). Although there is damping in the model, and the modes are actually complex, real modes may be calculated to estimate the resonances. The first 50 modes ranged in frequency from 1.87 Hz to 174 Hz, with the three members experiencing different types of motion in each mode (i.e., axial, flexural, or torsional), showing the need for the calculation of all types of power flows in beams.

The end load was then applied for frequencies ranging from 1 to 100 Hz, with a resolution of 1 Hz. The plot shown in Fig. 6 shows the response of section three at 100 Hz, or the 29th mode of the truss. The right end of the plot is the loading point, and the left end is the junction with Section 1 and the mounting at the bottom. Both power flow and power dissipation are plotted. Power flow decreases as it propagates along the beam due to power dissipation. Power dissipation oscillates from low to high points, approximating the mode shape of the beam. When dissipation is large, power flow slopes downward; when dissipation is small, power flow remains level.

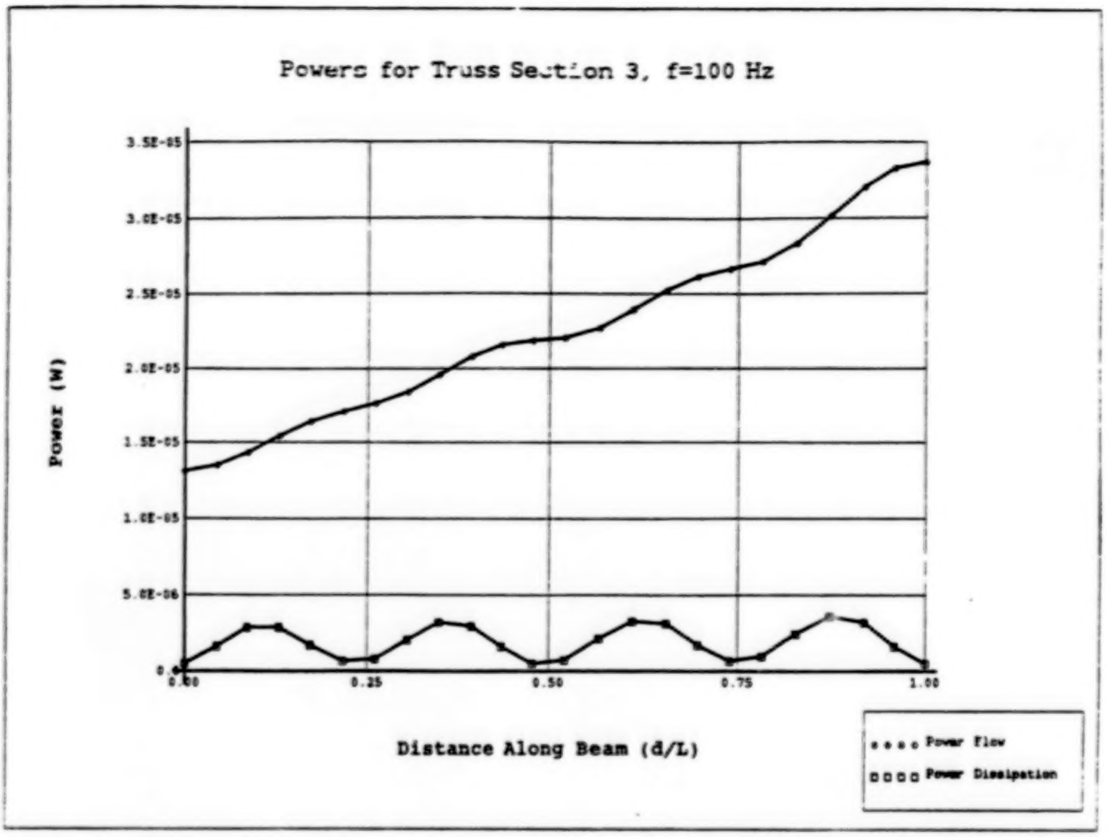


Fig. 6. Power Flows and Dissipations for a Single Frequency

The type of plot shown in Fig. 6 is an effective method of displaying the power flow response for a specific case; however spectra plots are required to illustrate the responses over the entire frequency range. An additional set of plots is shown in Fig. 7 and consists of four plots showing power flow at different locations on each of the truss sections. Section one is split into two graphs: graph one is for the top half of the beam, and graph two is for the bottom half.

Power flows are plotted for three distances along each member: at the beginning, middle, and end (d/L = 0.0, 0.5, and 1.0 respectively). For the top and bottom halves of Section 1, the beginning of the section is at the joint with

BEST COPY
AVAILABLE

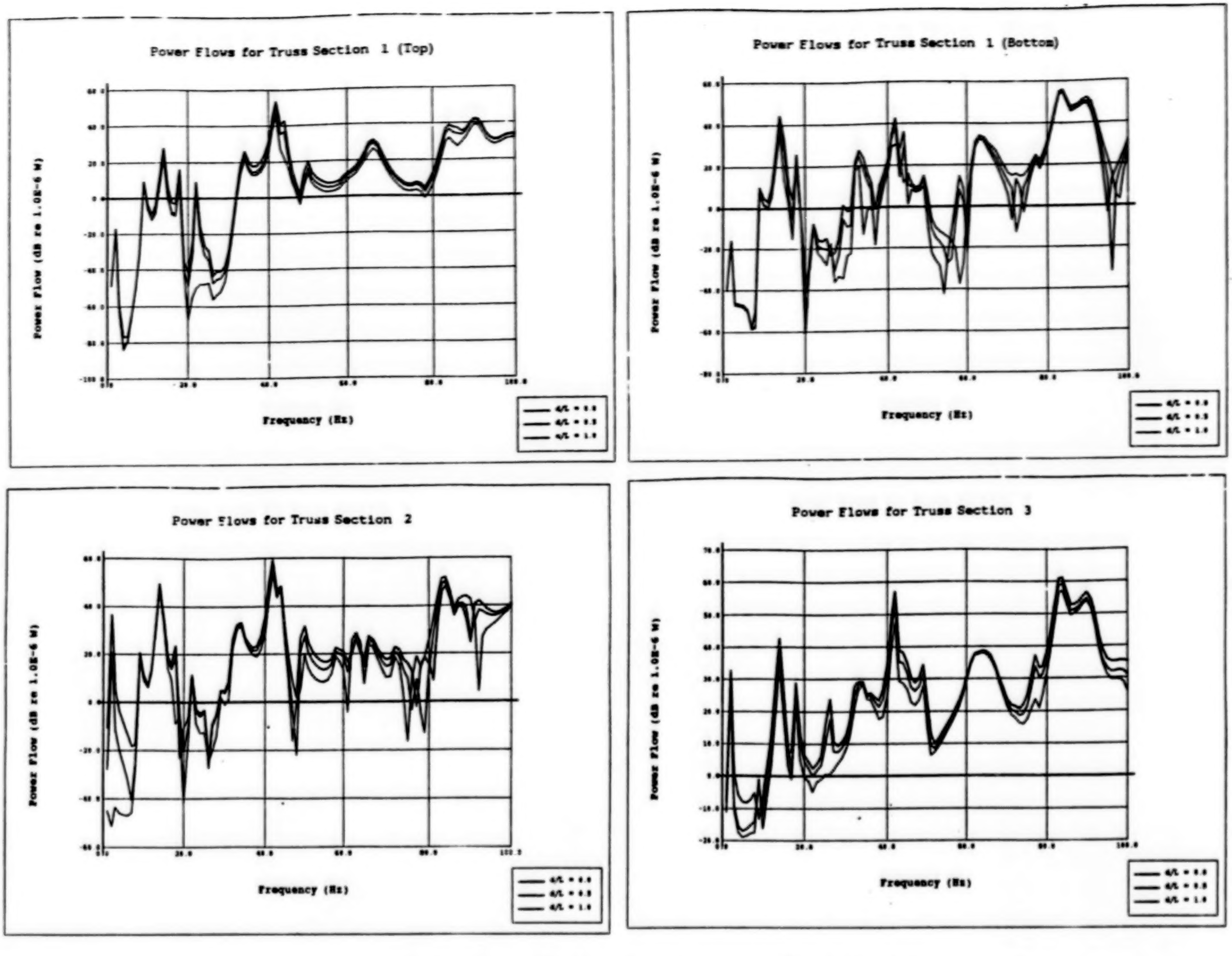


Fig. 7. Spectrum Plots of Power Flows for Truss Sections.

BEST COPY
AVAILABLE

Section 2; for Sections 2 and 3, the beginning of the member is at the load point.

The expected response is a relatively uniform lowering of each curve as power flow progresses from beginning to end along each beam. This is indeed the case for some frequencies. However, at some joints, such as the junction of Sections 1 and 2, and the junction between Sections 1, 3, and ground, power flows in ways that are less intuitive. As a result, some of the plots "cross over" each other and power flow increases from beginning to end. Fig. 8 contains three power flow diagrams which show some of the ways that power may flow through the truss model in this analysis.

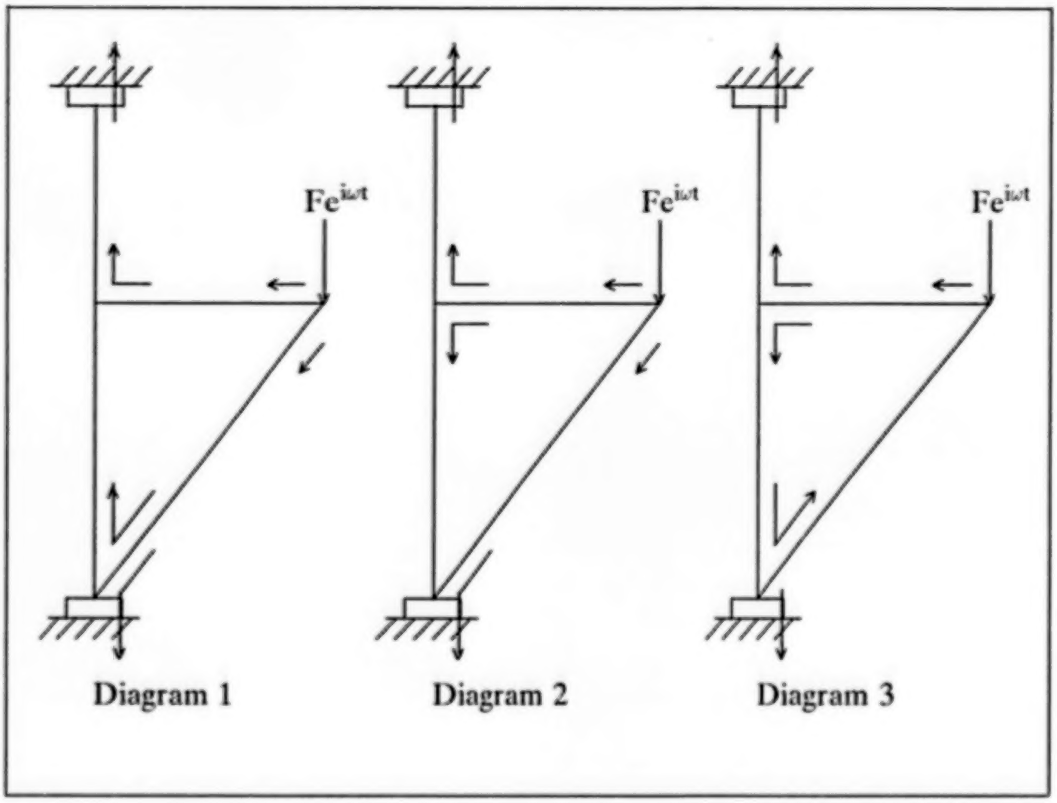


Diagram 1 Diagram 2 Diagram 3

Fig. 8. Power Flow Diagrams for Truss Problem

Diagram 1 shows power entering Sections 2 and 3 at the load points and flowing out toward Section 1. At the junction of Sections 1 and 2, the power flows from Section 2 into the upper half of Section 1. At the junction of Sections 1 and 3, power flows from Section 3 into the scalar elements connected to ground and into the bottom half of Section 1. Power then flows from the bottom of Section 1 to the top of Section 1, where it then flows up to the scalar elements at the top and out of the model.

BEST COPY
AVAILABLE

Diagram 2 shows a similar case, but with two differences. At the junction of Sections 1 and 2, power from Section 2 flows into the top and bottom halves of Section 1; and at the junction of Sections 1 and 3, power flows only into ground.

In Diagram 3, power is input only into Section 2, and flows into the top and bottom halves of Section 1. The power in the bottom half of Section 1 flows down to the junction of Section 1, Section 3, and ground, where some power flows out of the model and some flows up into Section 3.

The cases shown in Diagrams 1 and 2 are the most common based on examination of the printed output. Other possibilities exist, but do not occur often for the range of frequencies analyzed. The type of power flow diagram which occurs for a given frequency may be found by looking at the plots in Fig. 7. When power flow increases travelling from d/L = 0.0 to d/L = 1.0, then power has entered the beam at d/L = 1.0. When power flow decreases travelling from d/L = 0.0 to d/L = 1.0, then power has entered the beam at d/L = 0.0.

Response peaks in the graphs shown in Fig. 7 correspond to different types of motion in each section. Some peaks represent flexural motion, some are due to axial response, and some are torsional in nature. A power flow algorithm which considers only flexural response would give incorrect answers to this problem.

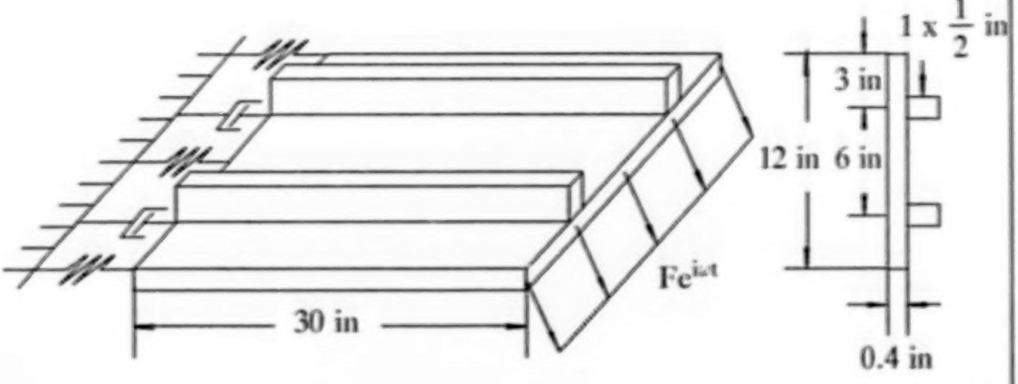
Power balances ($P_{in} = P_{out} + \sum P_{diss}$) were reasonably accurate across the frequency band, with small errors at frequencies of low response. It is uncertain which quantities are in error (P_{in}, P_{out}, or P_{dis}) for these cases, however the errors are of little consequence with respect to the calculations at higher responses. Power flows at the truss joints balanced as well. A calculation similar to Kirchoff's current law can be made, with power flows in the BARs connected to the junctions analagous to currents.

Beam-Stiffened Cantilever Plate

The analysis of ribbed stuctures combines the power flow methods for beams and plates. Nilsson[23] used SEA methods to predict the transmission of structure-borne sound through ribbed plate models. Here, FEA is used to calculate the low frequency response of a beam stiffened cantilever plate.

Problem Statement

A diagram of the model is shown in Fig. 9. Similar to the truss model, the cantilever plate model was attached to ground at its end by springs and dampers in all six DOF. The scalar elements simulated the effects of fasteners and the surrounding structure(s). A uniform end load was applied in the axial, transverse shear, and bending directions. A 12 x 30 mesh of QUAD2 elements was used to model the plate and two sets of 30 BAR elements modeled the stiffeners. The BAR elements were offset relative to the plates.



Material: Steel
$E = 30 \times 10^6$ psi
$\rho = 7.324 \times 10^{-4}$ lbs/in^4
$\nu = 0.3 \quad \eta = 0.02$

(Drawing not to scale)

Fig. 9. Beam-Stiffened Cantilever Plate Problem

For the scalar elements, spring constants were set at about 100 to 1000 times the stiffness of the members at the appropriate DOF; and the damping constant was set at ten times the material damping constant, or 0.2.

This model illustrates the power flow capability for plate elements, and helps further test the beam element formulation. Also, the power balance equation is checked for the case of multiple element types in a model; the total power dissipation in the beams plus the total power dissipation in the plates must match the difference of power input and power output.

Results

An eigenvalue extraction of the model showed the first 25 natural frequencies ranging from 15 to 2,122 Hz. Loads were applied to the model for a frequency range of 15 to 465 Hz with a resolution of 15 Hz.

A plot of power flows in one of the beam stiffeners over the frequency range is shown in Fig. 10. Since the model and the loading function are symmetric about the center of the plate, power flows through both beam stiffeners are the same. Curves are graphed for three locations along the beam, with d/L = 0.0 at the load point. Only two significant resonances appear in the plot; one peak occured at 75 Hz, and the other at about 240 Hz. Although the eigenvalue analysis of the problem predicts other resonant frequencies in this analysis range, their effects are likely felt in the plate section of the model.

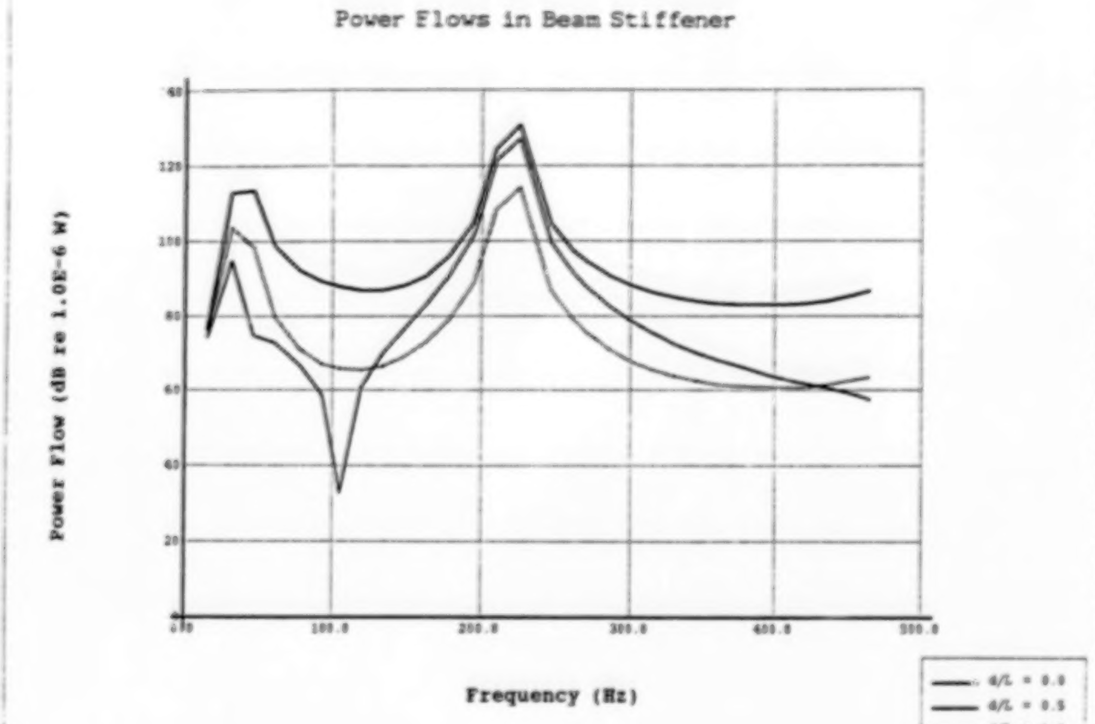


Fig. 10. Power Flows for Three Locations Along Beam Stiffener

For a plate element problem, spectrum plots are more difficult to generate and understand. Graphical (contour and vector) plots are needed to show the spatial variation of the power flow variables. A contour plot of the power flow magnitudes of the plate and beam elements is in Fig. 11. The beam elements are illustrated as plates in the diagram so their results may be visualized. Fig. 11 shows how power flows through the model at 455 Hz. Power flows into the model at the load points at the end of the plate, where some of it channels down the beam stiffeners, and the rest flows through the plate.

Fig. 12 shows a vector plot of power flow, which shows the directions that the power is flowing. The lengths of the arrows shorten as power travels from load point to the mountings at the end of the plate. In this case, almost all the power dissipated is due to material damping.

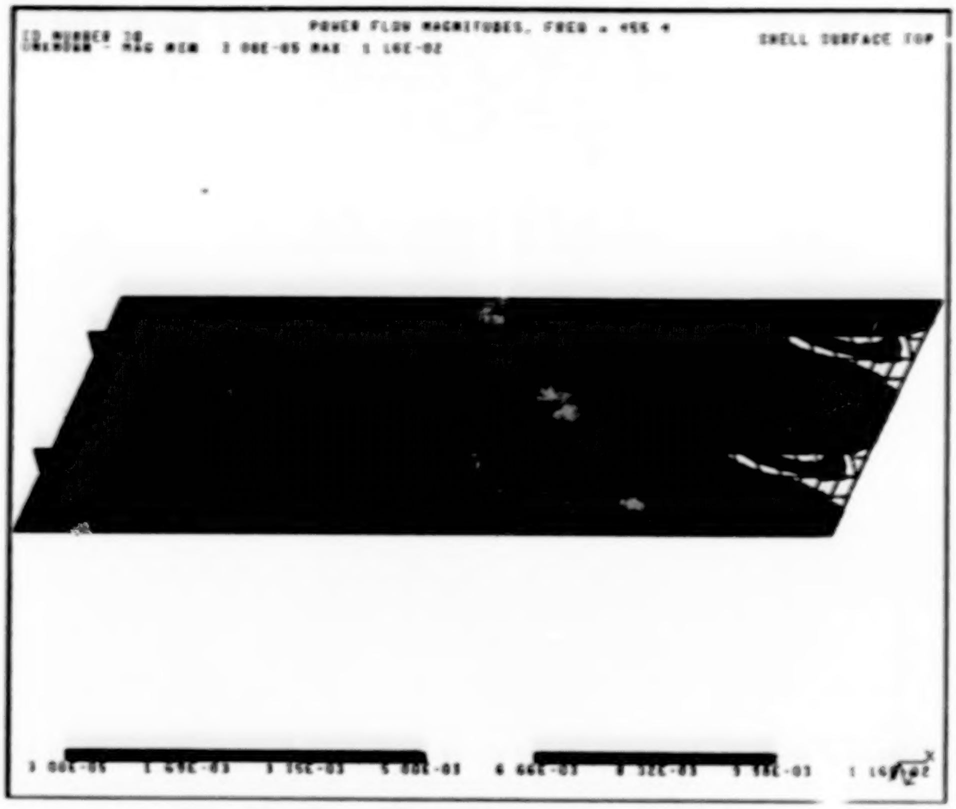


Fig. 11. Power Flow Magnitudes, f=455 Hz.

The effects of material damping are shown in Fig. 13, which is a plot of power dissipation. As mentioned in the "Damping and Power Dissipation" section earlier, a power dissipation field will resemble a mode shape, since dissipations are directly related to the squares of the displacements. In this case, the largest power sinks are outside the beam stiffeners and toward the rear of the plate.

Power balances are reasonably accurate for all frequencies, with the total power dissipations of the beam and plate elements matching the differences between input and output powers. The results show that both element types may be used accurately in a single model.

SUMMARY AND FUTURE WORK

A general capability for the calculation of power flow variables (power flow, mechanical intensity, power dissipation, power input and power output) has been developed for use with the finite elememt code NASTRAN. BAR, QUAD2, QUAD4, HEXA2, IHEXi, MASSi, and ELASi element types are currently supported. Unlike most of the studies presented in the literature, all types of power flows, flexural, axial, and torsional, are considered in the

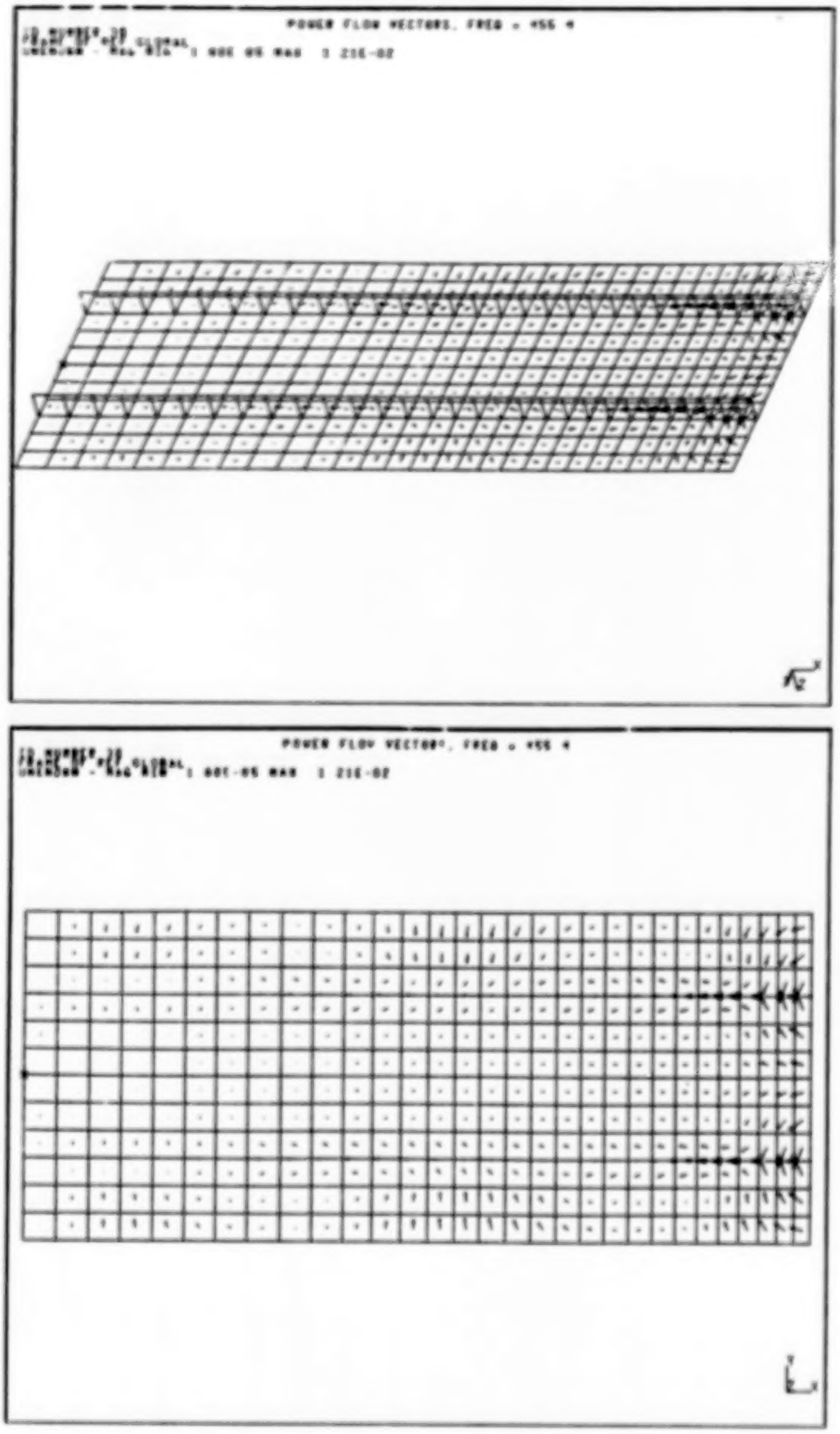

POWER FLOW VECTORS, FREQ = 455 H
POWER FLOW VECTORS, FREQ = 455 H

Fig. 12. Power Flow Directions (two views), f=455 Hz..

BEST COPY
AVAILABLE

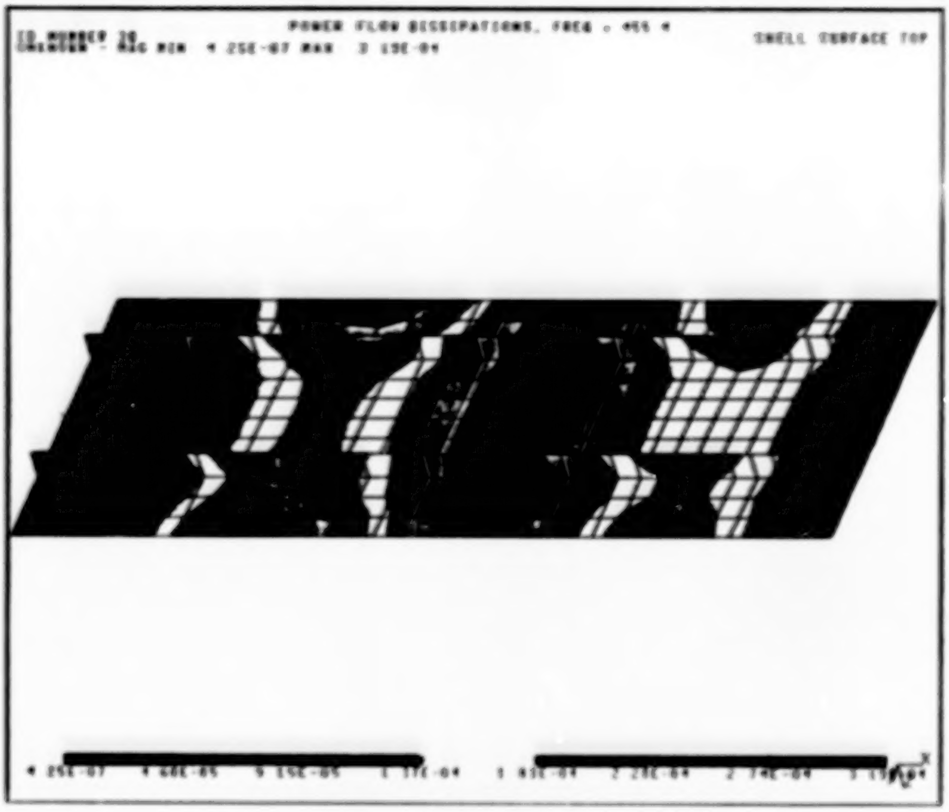


Fig. 13. Power Dissipations, f=455 Hz.

element formulations.

The results of the test problems indicate the method is a valid way of predicting the power flow response of a dynamically excited system at relatively low frequencies. Results for the test problems were more accurate at resonances than between resonances. Inaccuracies in the off-resonant responses are due to numerical problems; however, errors at low response are not as critical as errors at peaks.

Using FEA to calculate power flows is accurate and economical for the lower modes of a mechanical system. However the power flow results will only be as good as the NASTRAN results. Good modeling techniques and an understanding of the wavelength sizes of a problem are required. The shorter the wavelengths, the denser the required mesh will be.

Future work is extensive, and includes calculating power losses due to radiation damping, attaching dampers and active control devices to structures and measuring their effects, applying shape optimization techniques to structures with power flow variables as design constraints, determining the effects of mesh dependence, adding the power flow capability to NASTRAN, and developing a more specialized graphical post-processing package.

BEST COPY
AVAILABLE

REFERENCES

1. Wohlever, J.C., and R.J. Bernhard, "Vibrational Power Flow Analysis of Rods and Beams," Report No. 0353-12 HL 88-24, Ray W. Herrick Laboratories, Purdue University, West Lafayette, Indiana (1988).
2. Luzzato, E., and E. Ortola, "The Characterization of Energy Flow Paths in the Study of Dynamic Systems using S.E.A. Theory," *Journal of Sound and Vibration,* Vol. 123, No. 1, pp. 189-197 (1988).
3. Rasmussen, G., "Structural Dynamic Measurements using Intensity Methods," Proceedings of the Third International Modal Analysis Conference, pp. 558-564 (1985).
4. Rasmussen, G., "Transducers for Structure Intensity Measurements," Proceedings of Inter-Noise '87, pp. 1183-1186 (1987).
5. Pavic, G., "Measurement of Structure-Borne Wave Intensity, Part 1: Formulation of the Methods," *Journal of Sound and Vibration,* Vol. 49, No. 2, pp. 221-230 (1976).
6. Pinnington, E.J., "Vibrational Power Transmission to a Seating of a Vibration Isolated Motor," *Journal of Sound and Vibration,* Vol. 118, No. 3, pp.515-530 (1987).
7. Pavic, G., "Structural Surface Intensity: An Alternative Approach in Vibration Analysis and Diagnosis," *Journal of Sound and Vibration,* Vol. 115, No. 3, pp. 405-422 (1987).
8. Lyon, R.L., *Statistical Energy Analysis of Dynamical Systems,* The M.I.T. Press, (1975).
9. Mickol, J.D., and R.J. Bernhard, "An Investigation of Energy Transmission Due to Flexural Wave Propagation in Lightweight, Built-Up Structures," Report No. 0353-4 HL 86-40, Ray W. Herrick Laboratories, Purdue University, West Lafayette, Indiana (1986).
10. Cuschieri, J., "Power Flow as a Complement to Statistical Energy Analysis and Finite Element Analysis," *Statistical Energy Analysis,* ASME, NCA-Vol. 3, edited by K.H. Hsu, D.J. Nefske, and A. Adnan (1987).
11. Nefske, D.J., and S.H. Sung, "Power Flow Finite Element Analysis of Dynamic Systems: Basic Theory and Application to Beams," *Statistical Energy Analysis,* ASME, NCA-Vol. 3, edited by K.H. Hsu, D.J. Nefske, and A. Adnan (1987).
12. "NASTRAN® User's Manual, NASA SP-222(08)," Computer Software Management and Information Center (COSMIC), University of Georgia, Athens, Georgia (1986).

13. Cremer, L., M. Heckl, and E.E. Ungar, *Structure-Borne Sound*, Springer-Verlag, New York (1973).

14. Noiseux, D.U., "Measurement of Power Flow in Uniform Beams and Plates," *Journal of the Acoustical Society of America*, Vol. 47, No. 1, pp. 238-247 (1970).

15. Verheij, J.W., "Cross Spectral Density Methods for Measuring Structure-Borne Power Flow on Beams and Pipes," *Journal of Sound and Vibration*, Vol. 70, No. 1, pp. 133-139 (1980).

16. Li, J., "The Propagation of the Bending Waves and Torsional Waves in a Combined Beam Structure," Proceedings of Inter-Noise '87, pp. 611-614 (1987).

17. Williams, E.G., H.D. Dardy, and R.G. Fink, "A Technique for Measurement of Structure-Borne Intensity in Plates," *Journal of the Acoustical Society of America*, Vol. 78, No. 6, pp. 2061-2068 (1985).

18. Koshiroi, T., and S. Tateishi, "Visualization of Vibration Energy Flow in Plates: Measurements of Vibrational Intensity," Proceedings of Inter-Noise '87, pp. 1375-1378 (1987).

19. Fahy, F.J., and R. Pierri, "Application of Cross-Spectral Density to a Measurement of Vibration Power Flow Between Connected Plates," *Journal of the Acoustical Society of America*, Vol. 62, No. 5, pp. 1297-1298 (1977).

20. Cuschieri, J.M., "Extension of Vibrational Power Flow Techniques to Two-Dimensional Structures," first annual report, grant number NAG-1-685 from NASA Langley Research Center, (Aug 1987).

21. "Supertab® Engineering Analysis Pre- and Post-Processing User Guide and Reference Manual," Structural Dynamics Research Corporation, Milford, Ohio (1986).

22. Heckl, M., "Examples of Structure-Borne Sound Visualization," Proceedings of Inter-Noise '87, pp. 603-606 (1987).

23. Nilsson, A.C., "Excitation and Propagation of Structure-Borne Sound in Ribbed Structures," Proceedings of Inter-Noise '84, pp. 547-552 (1984).

FINITE ELEMENT ANALYSIS OF A MICROMECHANICAL DEFORMABLE MIRROR DEVICE

T. J. SHEERER, W. E. NELSON AND L. J. HORNBECK
TEXAS INSTRUMENTS INCORPORATED, DALLAS TX.

ABSTRACT:

Texas Instruments has developed a monolithic spatial light modulator chip consisting of a large number of micrometer-scale mirror cells which can be rotated through an angle by application of an electrostatic field. The field is generated by electronics integral to the chip. The chip has application in photoreceptor based non-impact printing technologies. Chips containing over 16000 cells have been fabricated, and have been tested to several billions of cycles. Finite Element Analysis (FEA) of the device has been used to model both the electrical and mechanical characteristics.

INTRODUCTION:

The very high component density achieved in integrated circuits is well-known. Using the same processing techniques it is also possible to produce micromechanisms on a similar scale. Petersen (1) has described the manufacture and testing of extremely small silicon cantilevers and also lists several commercial applications of micromechanical devices.

The deformable mirror device (DMD) consists of a chip containing a large number of mirror cells as shown in Fig.(1). The cell consists of a 19um x 19 um aluminum mirror pivoted at two corners by 4.5 um beams. The mirror and its support structure are 0.375um in thickness while the beam is of nominal width 1.0 um and nominal thickness 0.08 um. 2.3 um below the structure are two address electrodes and two landing electrodes. To rotate the mirror through an angle, , a bias potential,ϕ_b*,is applied to the upper structure and landing electrodes while appropriate address potentials are applied to the address electrodes as shown in Fig.(2). The cell may be used as an optical lever to deflect a light beam in and out of the field of a projection lens as shown in Fig.(3). The effect is thus of being able to activate and deactivate a "pixel" at the very high speed of the DMD cell. The DMD is more fully described in (2) and (3). In an electronic printer using a xerographic type process the light is used to selectively dissipate charge on a transfer medium, prior to the charge being transfered to paper,allowing the selective deposition of "toner" material on the paper. The most common type of printer using this process is the laser printer, which requires in addition to the laser a complex optical system and a mechanically rotating mirror to 'scan' the transfer medium. Use of the DMD (Fig.(4)) is simpler, allowing use of a conventional* *incandescent source. Additionally, by variation of the "on" time of the cell, the DMD allows use of grey-scale printing. Mechanical analysis of such devices by FEA is not different from analysis of large structures, the only requirement being judicious choice of units for dimensions and mechanical properties to avoid faults due to arithmetic overflow or underflow. Electrostatic analysis is also relatively simple, as the equations of electrostatics are identical with those of heat transfer. By use of COSMIC NASTRAN in these applications it is possible to accurately model the behavior of a DMD and assess the effects of design modifications without the very considerable expense of a production run.*

EQUATIONS OF ELECTROSTATICS AND THEIR ANALOGS:

The analogy between electrostatics and steady-state heat transfer is exact, and NASTRAN's heat transfer capabilities can be used in the solution of electrostatic field problems without modification. In heat transfer we have:

$$\dot{q} = - k . \nabla T \qquad (1)$$

whereas in electrostatics the polarization, **D**, *is given by:*

$$\mathbf{D} = - \varepsilon . \nabla V \qquad (2)$$

Heat sources and sinks are equivalent to point charges and fixed temperatures are equivalent to fixed potentials. In electrostatics the potential gradient is refered to as the field, **E**. *There is no directly analogous term in heat transfer. The analogy between the different terms is listed in Table (1) below:*

TABLE 1: ANALOGY BETWEEN ELECTROSTATICS AND HEAT TRANSFER

HEAT TRANSFER	***ELECTROSTATICS***
TEMPERATURE	***POTENTIAL V***
TEMPERATURE GRADIENT	***ELECTRIC FIELD E***
HEAT FLUX	***POLARIZATION D***
HEAT SOURCE	***POINT CHARGE q***
CONDUCTIVITY	***PERMITTIVITY*** ε

The force exerted on a charge by an electric field is given by:

$$\mathbf{F} = q \cdot \mathbf{D} \qquad (3)$$

*where **D** is the electric polarization and q the charge. As the charge itself contributes to the field it is necessary to integrate the above expression to obtain a useful expression in terms of **E**, giving normalized force, or pressure, in vacuo:*

$$\mathbf{F} = 0.5\, \varepsilon_0 \mathbf{E} \cdot \mathbf{E} \qquad \mathbf{(4)}$$

where is the permittivity of free space. This expression allows calculation of the pressure exerted at an element face directly from the output of the finite element analysis. The charge distribution resulting from a potential distribution is obtainable from the single point constraint forces which in the case of heat transfer indicate heat sources and sinks. One important difference in behaviour between the electrostatic case and the thermal case is that a conducting material acts in electrostatics in a manner equivalent to an infinitely conductive material in heat transfer, so that all points on the conductor are at the same potential. This can be modelled by appropriate constraints in the NASTRAN input deck.

ELECTROSTATIC MODEL OF THE DMD:

*A 3-D model of the DMD cell was constructed using solid elements as shown in Fig.(5). Fig.(6) plots the z-component of the electric field ,**E**z, vs. x-coordinate at the center and edge of the model for a bias potential of -12V and address potentials of +5V and zero. Fig.(7) is a fringe plot of **E**z superimposed on the model. On the basis of the results obtained it was determined that fringeing effects were of relatively small significance and a series of 2-D models were made with mirror rotation angles form 1 to 9 degrees in the XY plane. Rotation of approximately 9.2 degrees was sufficient to bring the mirror in contact with the landing electrode. From each of these models The useful component of electric field, **E**y, was obtained as a function of location on the mirror. In increments corresponding to element size on the model, the force exerted on the mirror was calculated as a function of location using **E**y and the mirror's width as a function of location.Figs.(8-11) are fringe plots of potential V and field **E**y for two different rotation angles, Figs.(12-13) plot **E**y vs. location for different rotation angles for bias potentials of -12V and -16V, with an address potential of +5V. The choice of +5V allows use of standard (TTL or CMOS) logic electronics to control the DMD, with the bias potential being generated separately.*

MECHANICAL MODEL OF THE DMD:

The mechanical model of the DMD was constructed in order to verify the assumption that the device acted as a rigid body (the mirror) mounted on torsion rods. Using the measured dimensions of the device a model was constructed using plate and bar elements and subject to modal and static analysis. It was determined that deformation of the mirror plate was negligible, and that the torsion bars had three deformation modes of interest with frequencies of 56.2, 192 and 545 kHz. as plotted in Fig.(14). The higher modes involved plate deformation at very high frequencies. These results indicated that the elastic constants used were valid (experimentally the first mode was at approximately 54 kHz) and that in applying electrostatic forces to the mirror it was valid to use a single lumped force at an arbitrary node, and not necessary to apply pressures over each element of the DMD model. It was also determined from this model that a force of 1.74 millidynes applied at the DMD tip was required to fully rotated the mirror.

RECONCILIATION OF MODELS AND EXPERIMENTAL DATA:

Using the electrostatic DMD model it is possible to plot force or torque on the DMD against angle for different bias and address potentials. It should be possible to predict the potentials at which the device can fully rotate. In this circumstance the electrostatic force will exceed the mechanical restoring force at all angles of rotation, and will tangentially approach the restoring force curve at a critical angle which is readily determined experimentally. Experimentally, for an address potential of 5V, a bias potential of -16V is found necessary, with a critical angle around 5.5 degrees. Fig.(15) shows experimental data plotting bias voltage vs. address voltage for spontaneous rotation. Fig.(16) plots the net force on the DMD from electrostatic fields vs. angle for an address potential of +5V and bias potential of -12V and -16V. Also plotted is the mechanical restoring force based on the finite element structural model of the DMD. The curve for -16V does appear to tangentially approach the mechanical force curve at around 5 degrees, and thus is in excellent agreement with the experimental data.

CONCLUSIONS:

The use of very simple 2-D models of electrostatic fields has proven to give accurate values of electrostatic forces in a micromechanical device. It appears likely that any desired accuracy is attainable if a sufficiently complex model is used. The response time of the device, although fast for a mechanism, is much less than the rise time of the electrode potentials. Thus the data on force vs. rotation could also be used in a transfer function or nonlinear spring input to allow dynamic modelling of the device.

REFERENCES:

(1) Kurt E. Petersen, "Silicon as a Mechanical Material", Proc. IEEE, ***70-5****, 420, (1982)*

(2) L.J. Hornbeck, "128x128 Deformable Mirror Device", IEEE Trans. Electron Devices ***ED-30(5)****, 539, (1983)*

(3) D.R. Pape and L.J. Hornbeck, "Characteristics of the Deformable Mirror Device for Optical Information Processing", Optical Engineering, ***22(6)****, 675 (1983)*

FIG. (1)

BISTABLE DEFORMABLE MIRROR DEVICE

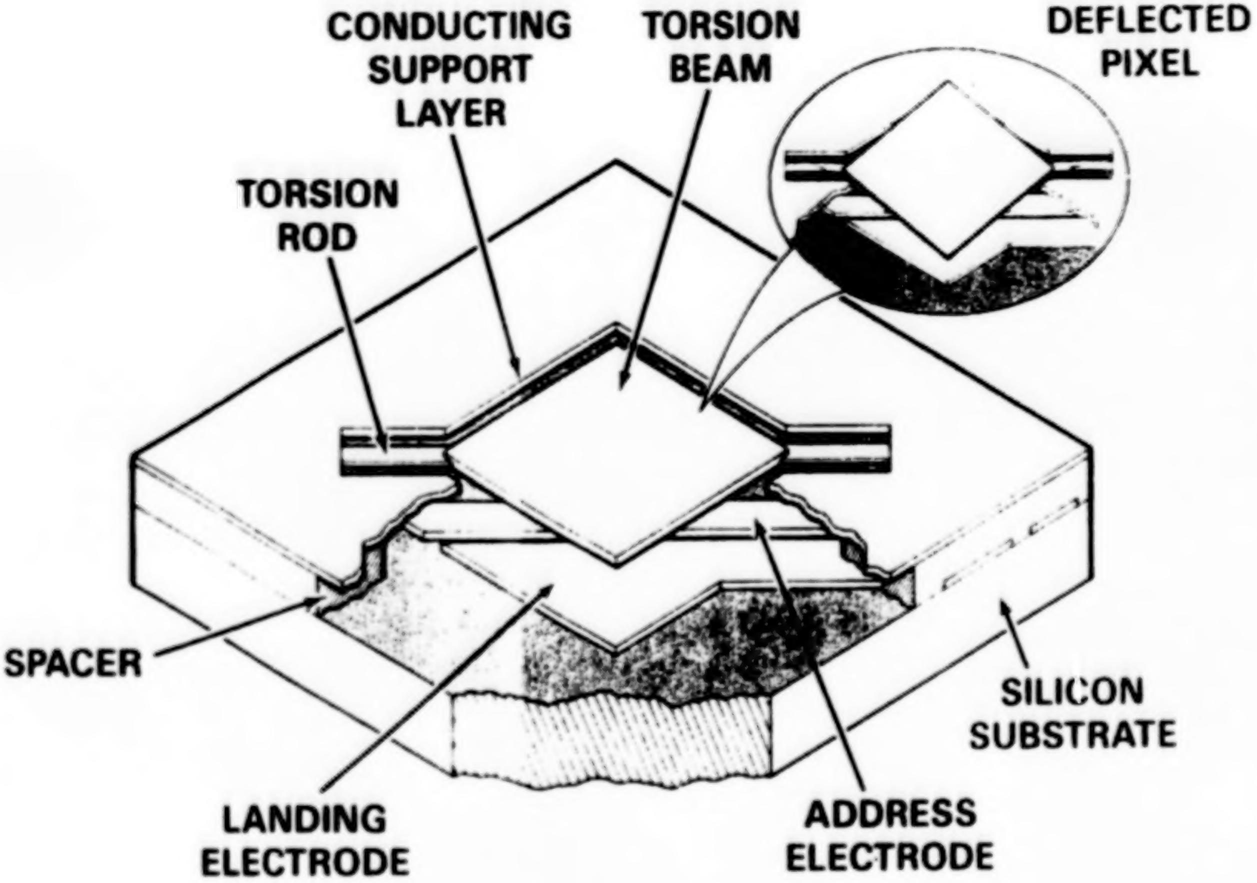
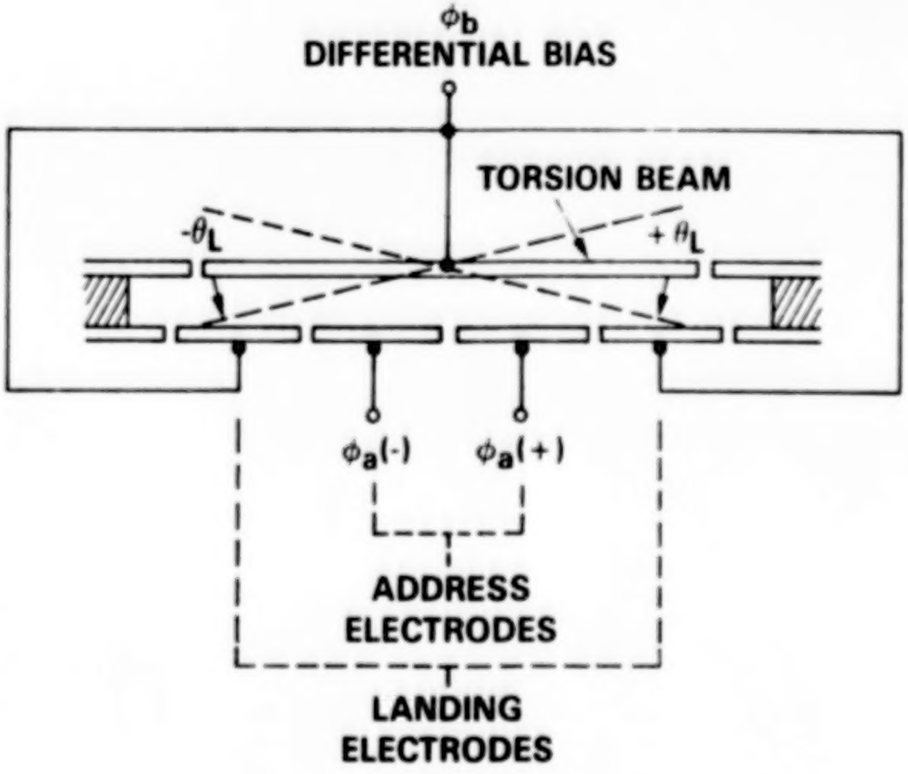
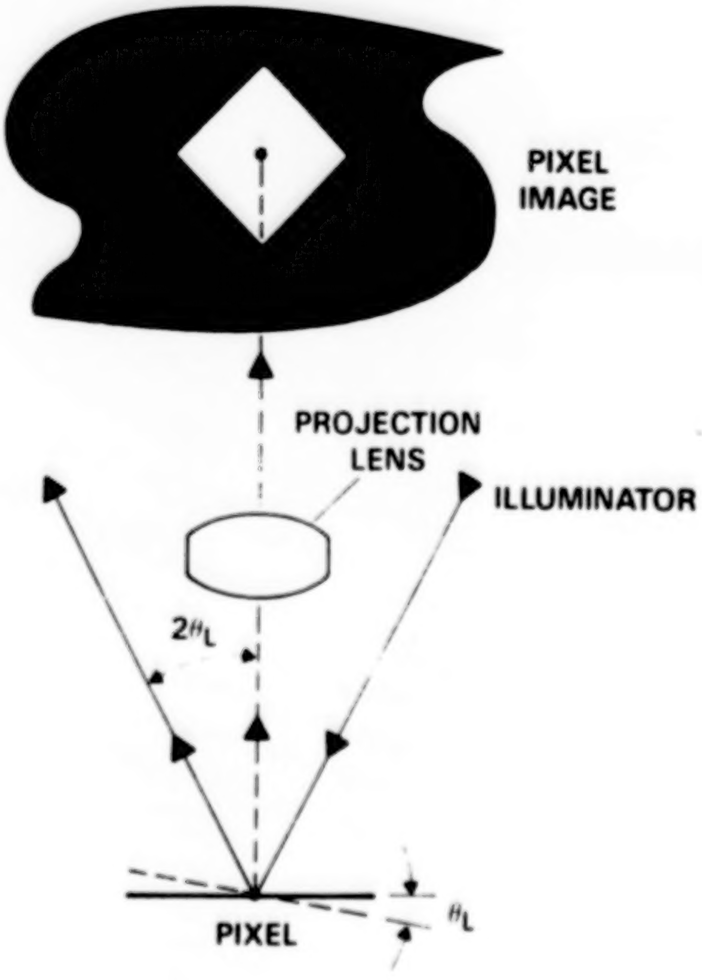

CONDUCTING SUPPORT LAYER
TORSION BEAM
DEFLECTED PIXEL
TORSION ROD
SPACER
SILICON SUBSTRATE
LANDING ELECTRODE
ADDRESS ELECTRODE

FIG. (2)

BISTABLE DMD
SCHEMATIC CROSS SECTION


ϕ_b
DIFFERENTIAL BIAS
TORSION BEAM
$-\theta_L$
$+\theta_L$
$\phi_a(-)$
$\phi_a(+)$
ADDRESS ELECTRODES
LANDING ELECTRODES



BISTABLE DMD
DARKFIELD PROJECTION

PIXEL IMAGE
PROJECTION LENS
ILLUMINATOR
$2\theta_L$
θ_L
PIXEL

FIG. (3)

FIG.(4)

DMD LIGHT MODULATOR SYSTEM WITH PRINTER

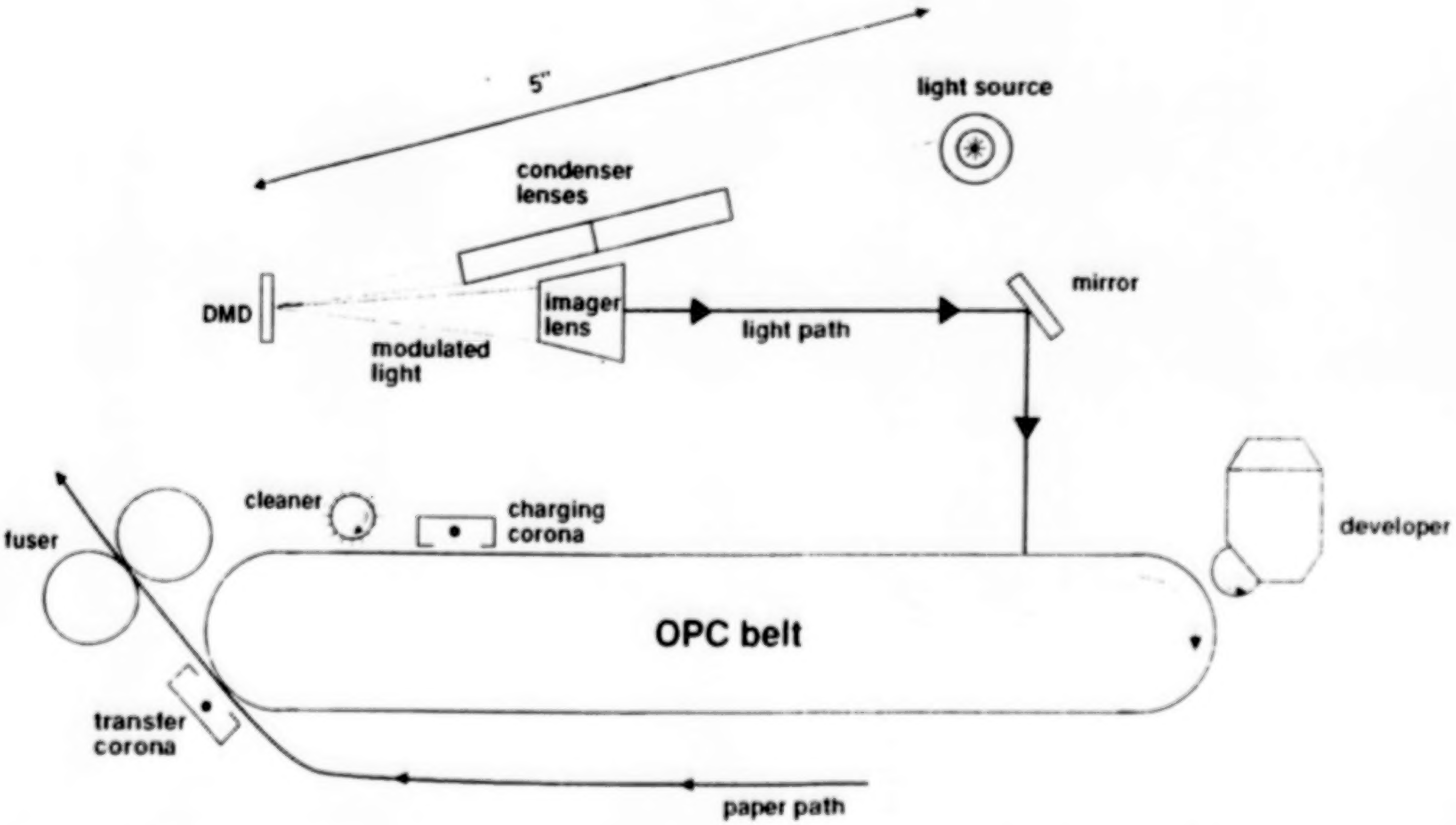
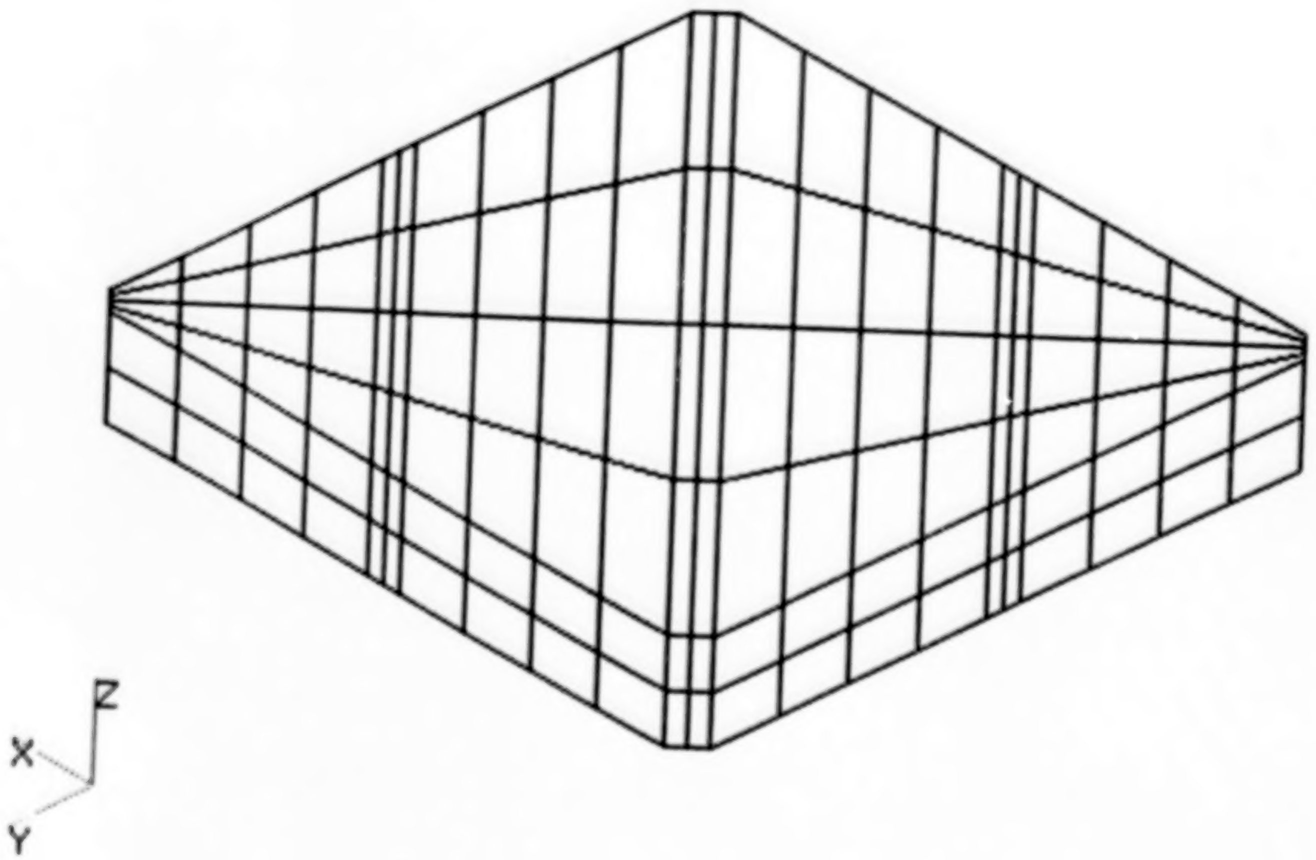

5"
light source
condenser lenses
DMD
imager lens
modulated light
light path
mirror
cleaner
charging corona
fuser
developer
OPC belt
transfer corona
paper path

FIG.(5), 3-D MODEL OF DMD



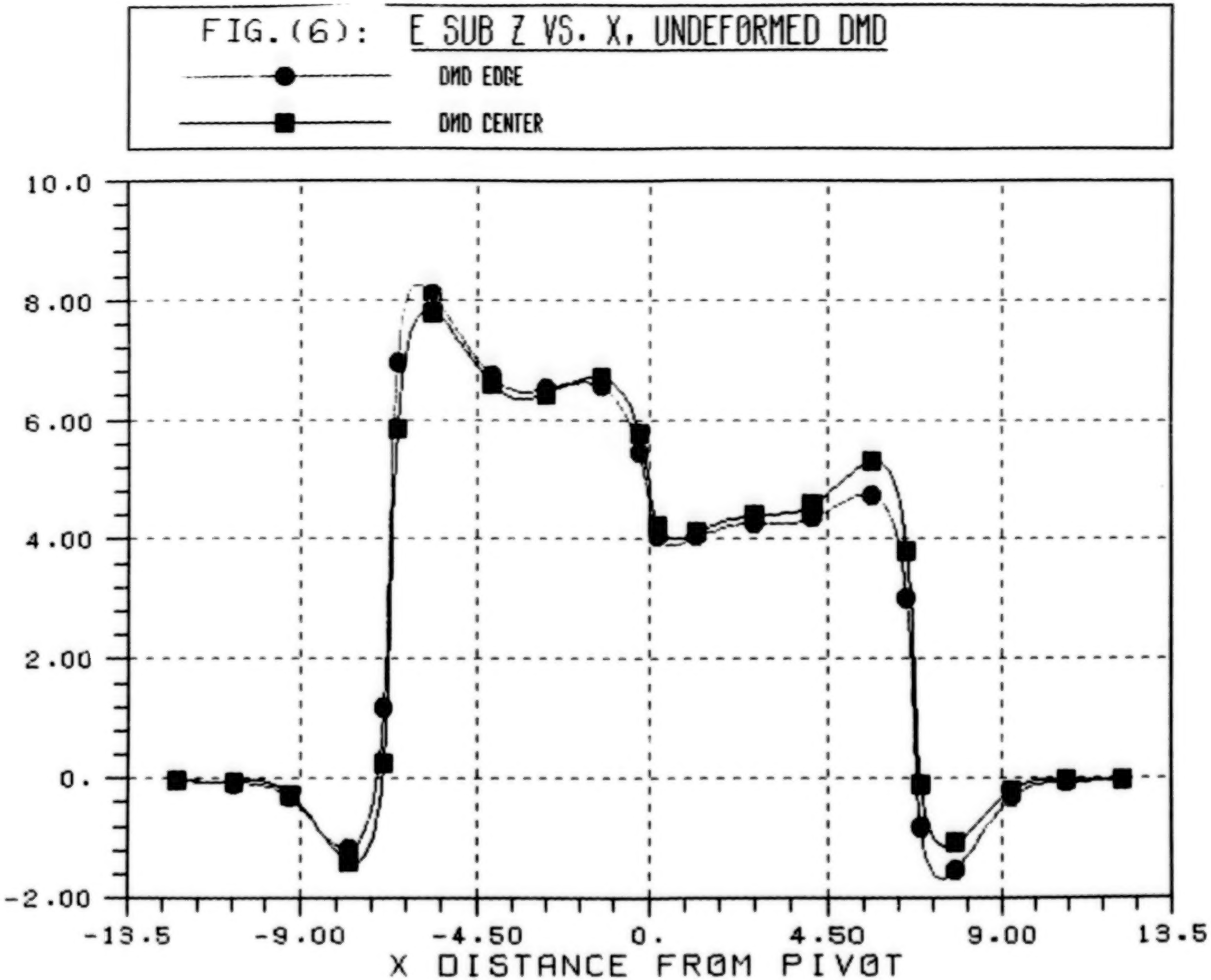


FIG.(6): E SUB Z VS. X, UNDEFORMED DMD

FIG(7), E SUB Z IN 3-D MODEL OF DMD

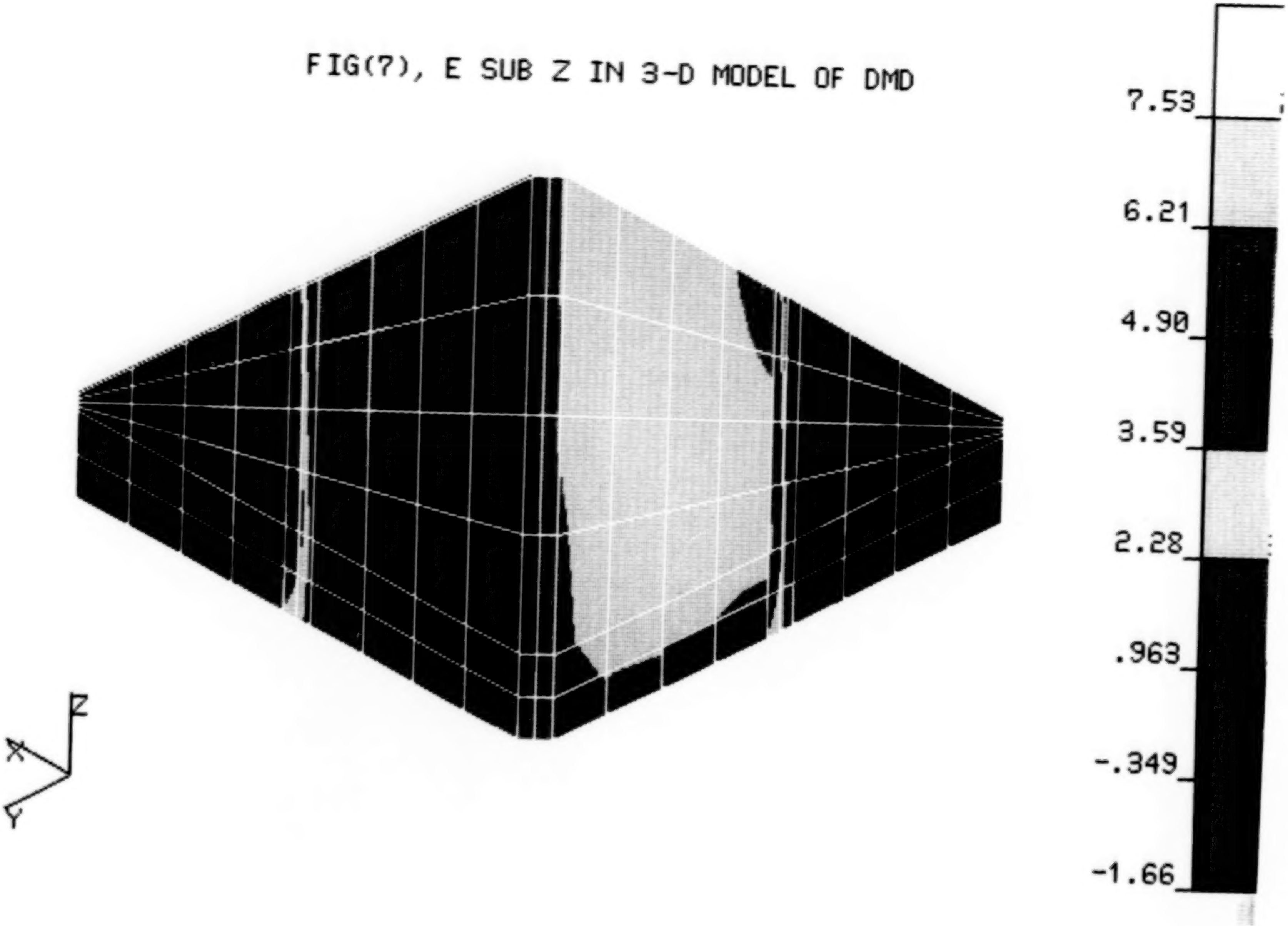
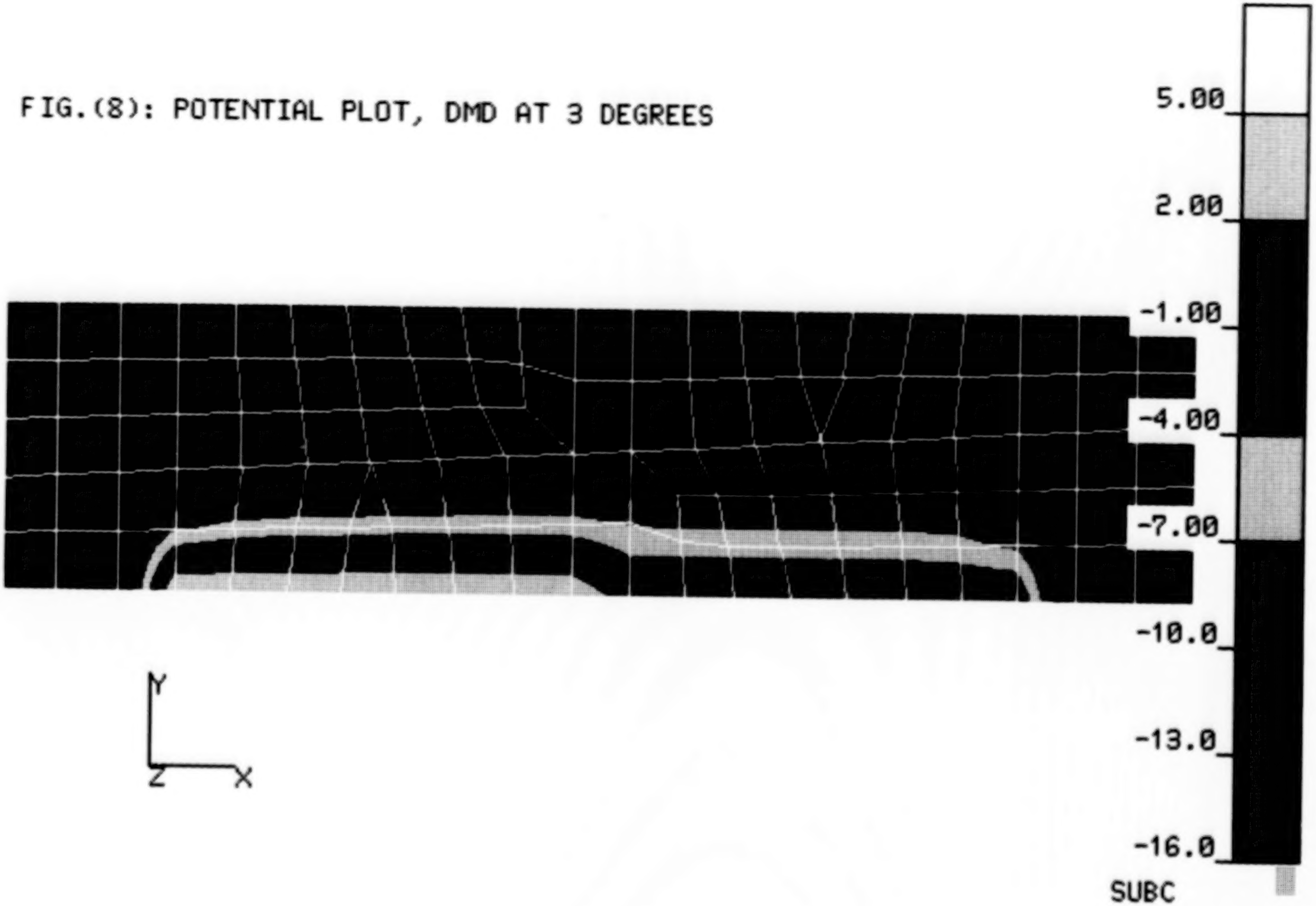


FIG.(8): POTENTIAL PLOT, DMD AT 3 DEGREES



FIG(9), E SUB Y, DMD AT 3 DEGREES

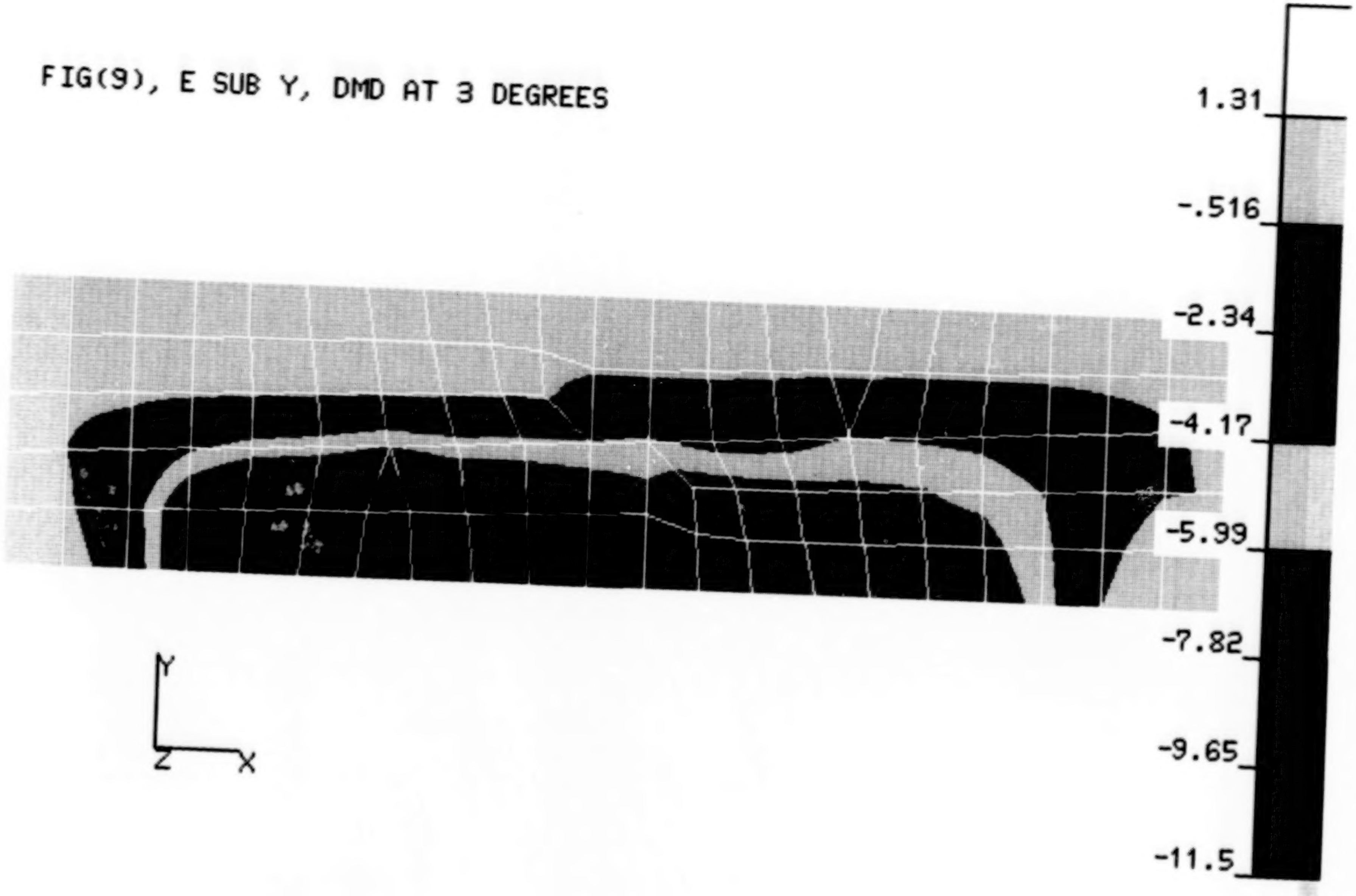
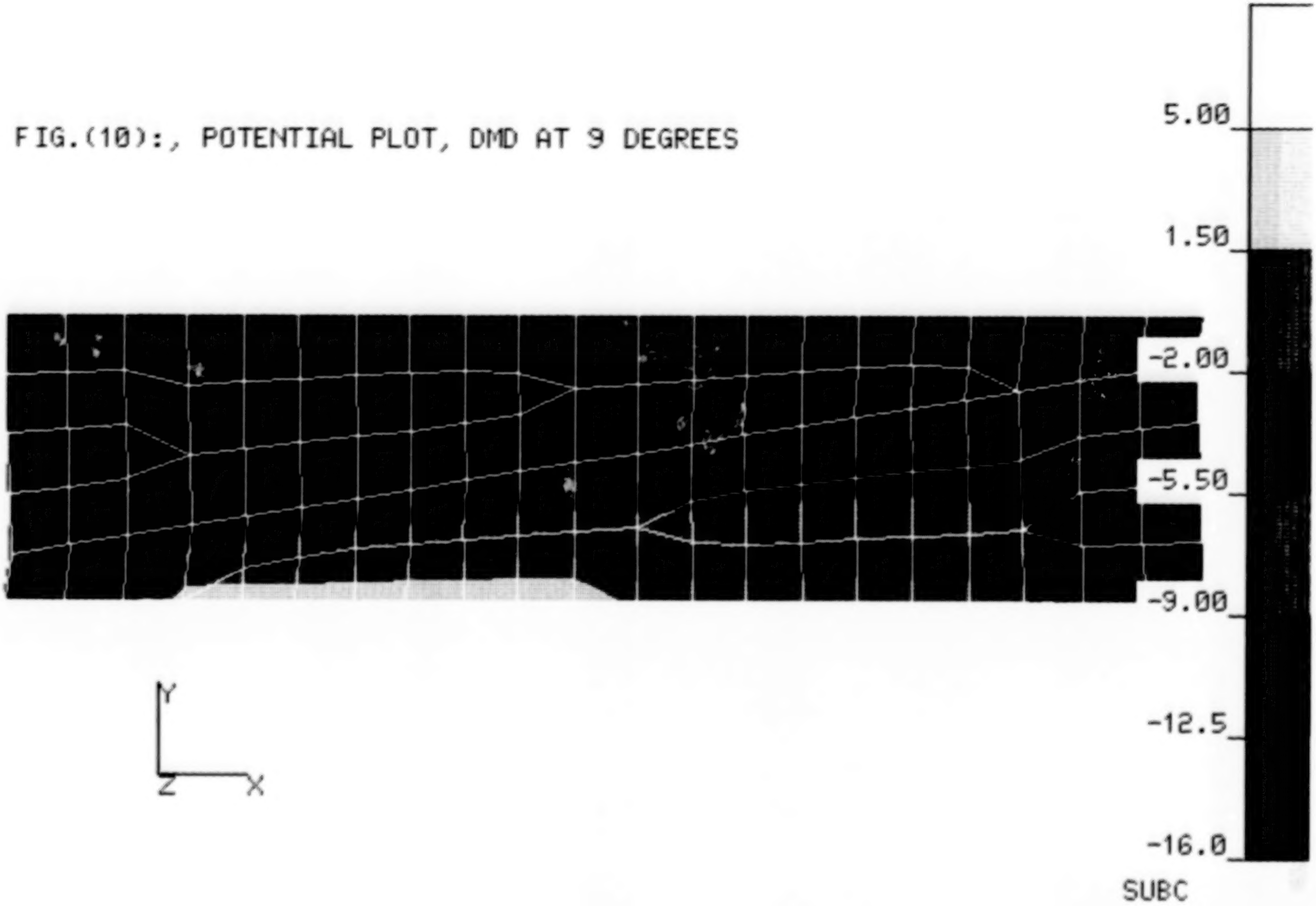
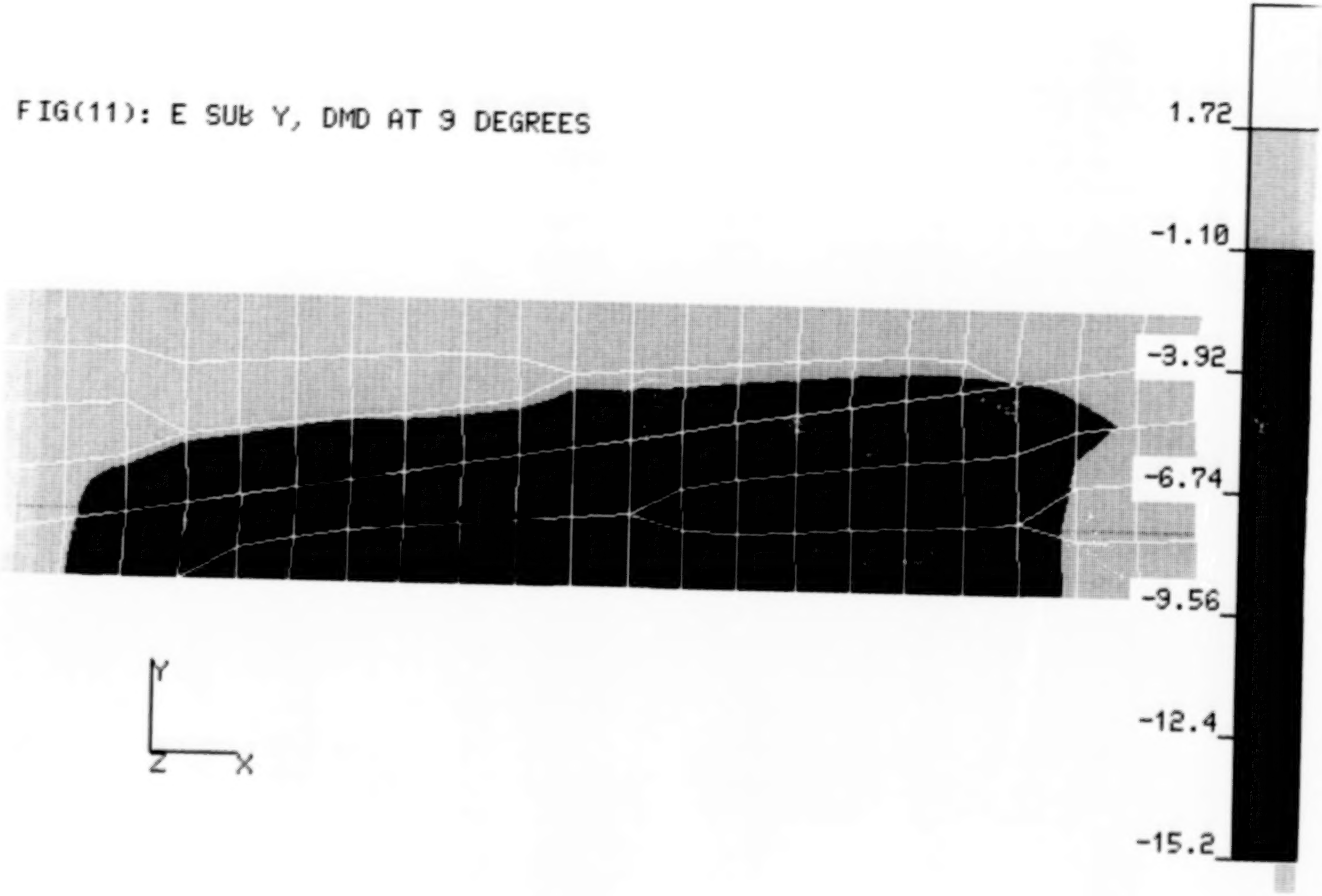


FIG.(10):, POTENTIAL PLOT, DMD AT 9 DEGREES



FIG(11): E SUB Y, DMD AT 9 DEGREES



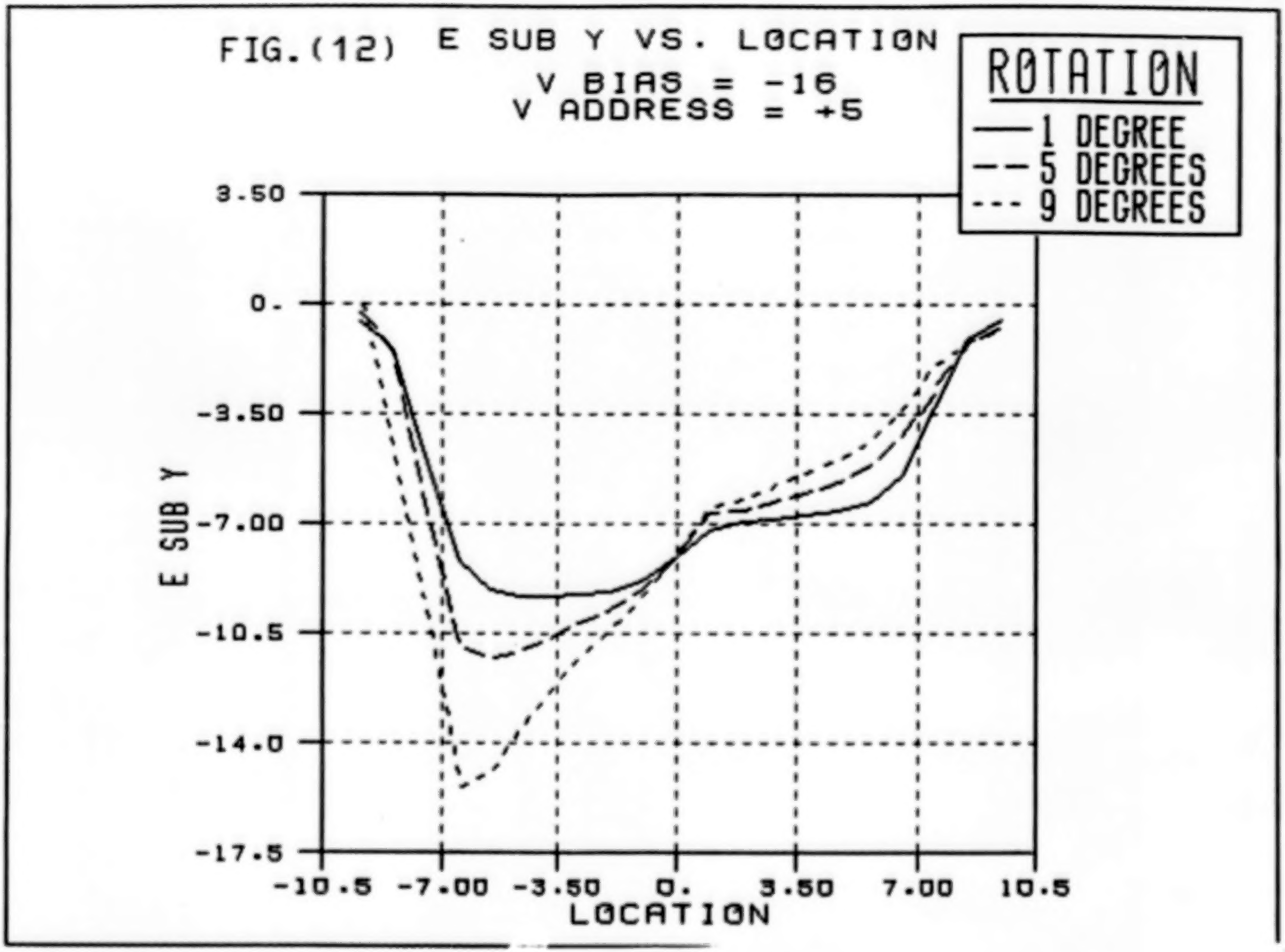


FIG.(12) E SUB Y VS. LOCATION
V BIAS = -16
V ADDRESS = +5

FIG.(13)

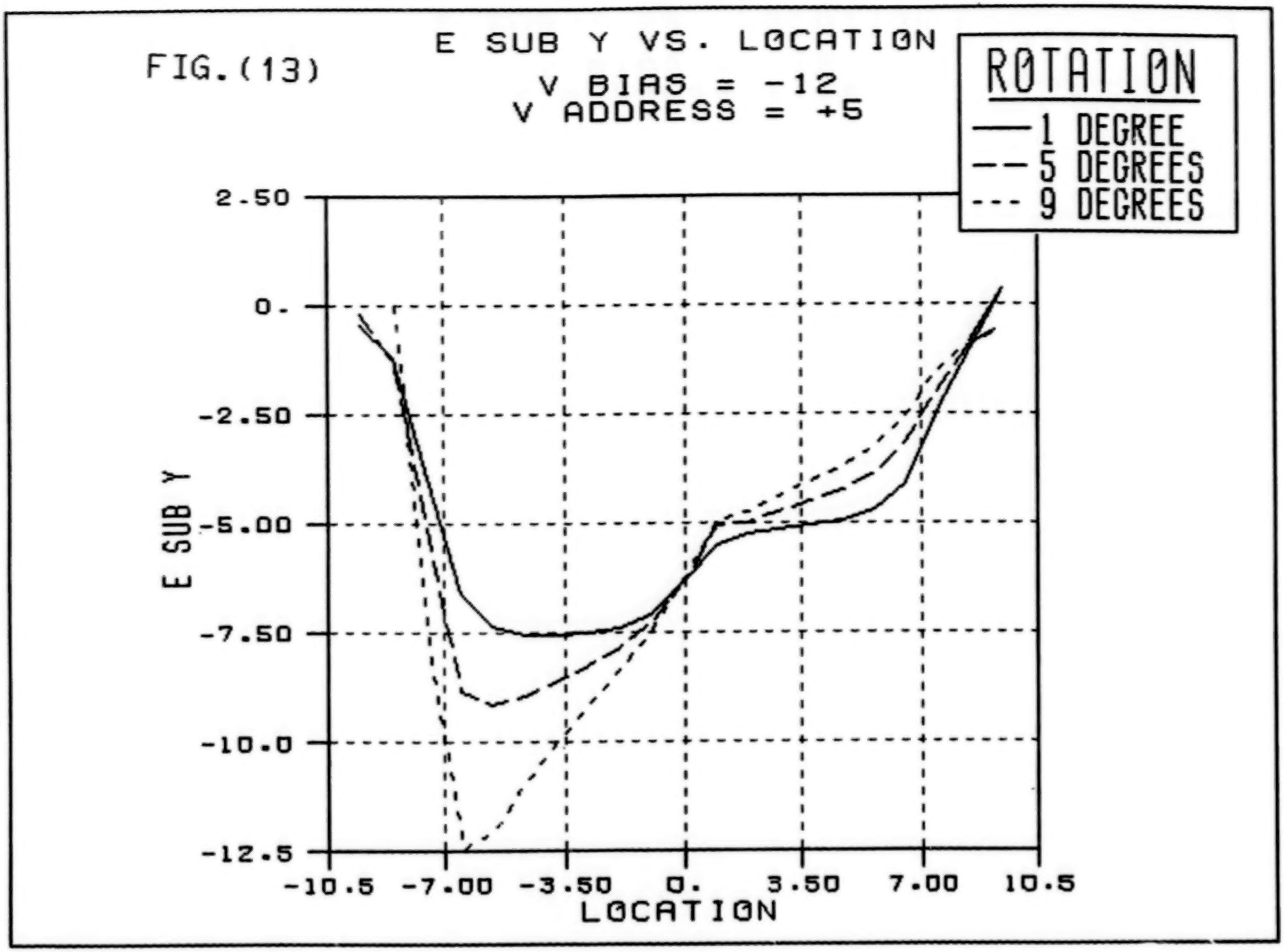

E SUB Y VS. LOCATION
V BIAS = -12
V ADDRESS = +5
ROTATION
1 DEGREE
5 DEGREES
9 DEGREES
E SUB Y
2.50
0.
-2.50
-5.00
-7.50
-10.0
-12.5
LOCATION
-10.5
-7.00
-3.50
0.
3.50
7.00
10.5

FIG.(14): DMD VIBRATION MODES

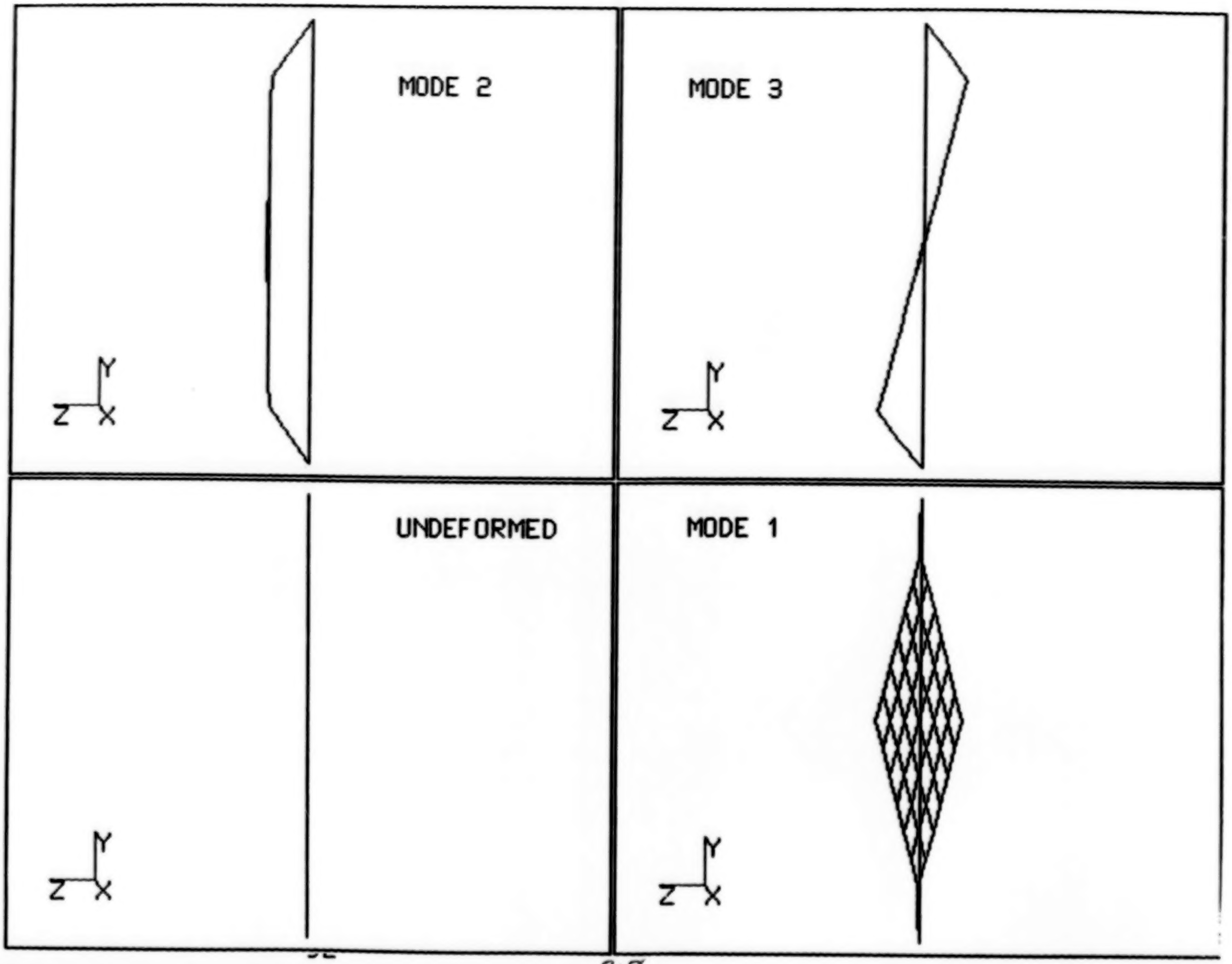

MODE 2
MODE 3
UNDEFORMED
MODE 1
Y
Z
X

FIG.(15)

BISTABLE DMD

DIFFERENTIAL BIAS VS ADDRESS VOLTAGE

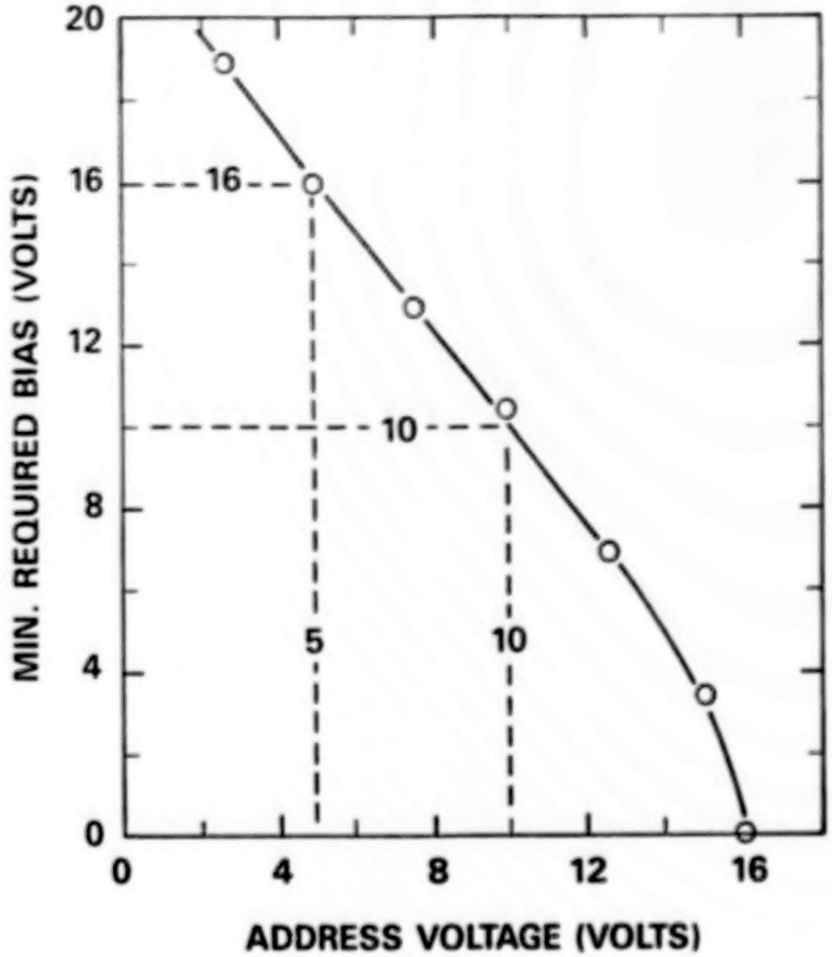
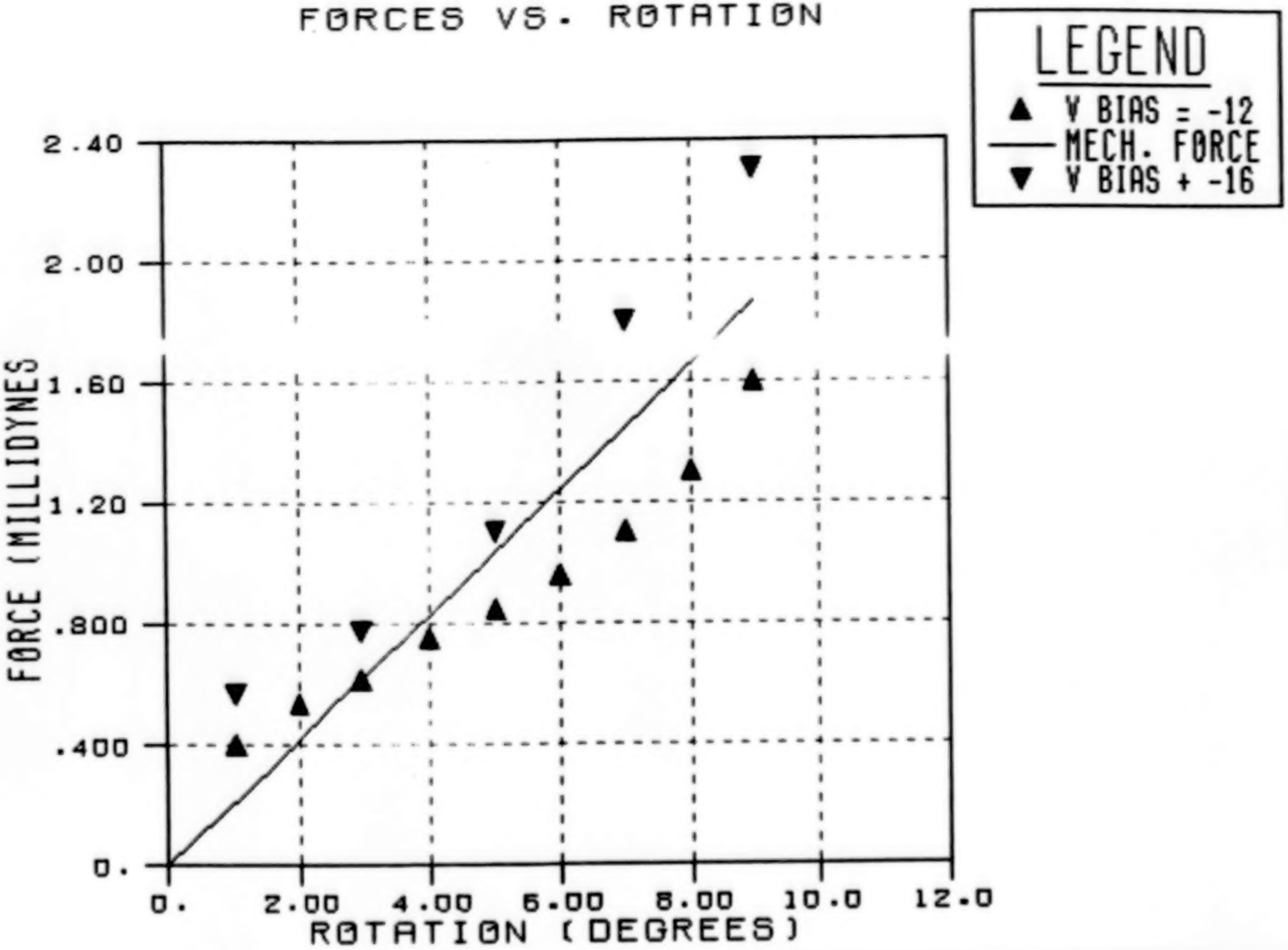


FIG.(16): ELECTROSTATIC AND MECHANICAL FORCES VS. ROTATION



RITZ METHOD FOR TRANSIENT RESPONSE IN SYSTEMS HAVING UNSYMMETRIC STIFFNESS

Thomas G. Butler
BUTLER ANALYSES

If the choice of generalized coordinates for determining the transient response of a non-symmetric structure were not eigenvectors but were modes of deformations due to operating loads, there would be certain advantages. Among these would be: 1. the economy of requiring only a small number of modes, 2. the avoidance of having either to cull out certain non-participating modes or to retain the non-participating modes at the expense of having to operate with larger order matrices, and 3. the confidence of getting well converged solutions. Using load response modes as generalized coordinates is properly classified as the Ritz Method.

The interest in the case of non-symmetric stiffness derives from structures with active control systems. The assymmetry comes from the sensor being situated at a different location than the actuator. Loads that would be typical of those used in the design of control systems are: externally applied forces and pressures, vernier jets whose firing is commanded by the control system, and constraints from appendages assigned to attach points. Such a solution method is developed here as a DMAP ALTER packet to the Statics Rigid Format in NASTRAN. The Inertia Relief Rigid Format would appear to be a more natural host for such an ALTER packet, because it carries out full processing of mass. Its prohibition against boundary constraints disqualifies it, hence Statics with modifications to process mass is used as host.

THEORETICAL APPROACH

The theory will be organized in 5 parts. It will describe the mathematical decisions on using the original stiffness matrices in the development of fundamentals first. It will describe the generation of harmonics from each fundamental, second. Then it will develop the adjoint vectors third. Having a full complement of primary and adjoint generalized vectors, the next step is to orthogonalize them and finally to integrate them into an actual solution by constructing the generalized mass and generalized stiffness and reconciling the form of the generalized damping.

The methods presented herein are an outgrowth of a new non-collocated sensor actuator analysis method under development by H. P. Frisch at the NASA/GSFC, Code 712.[1] The motivation for the NASTRAN/DMAP implementation presented herein is to provide a working capability which can be used for both current practical applications and for the evaluation of the new analysis method prior to its inclusion into the general purpose multi-flexible body data preparation program FEMDA.

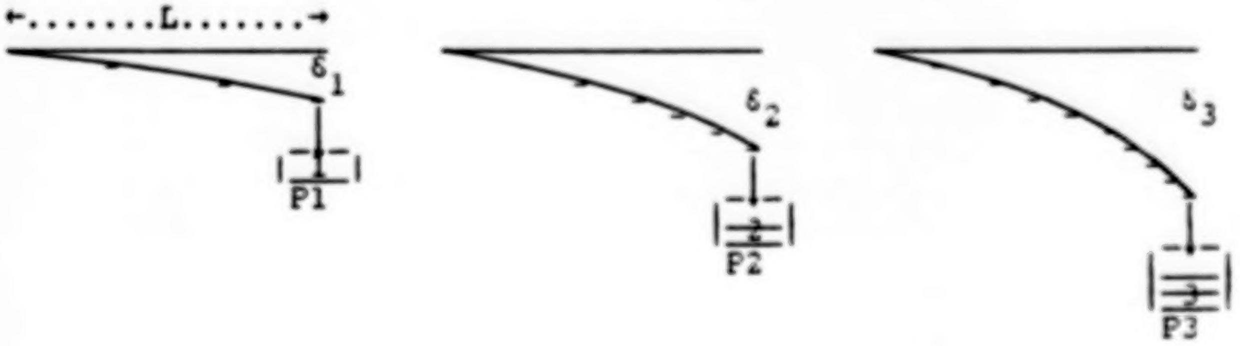
---

1. Frish, H.P. "IAC Program FEMDA, Theory and User's Guide, Interface from Structural Analysis Output Data to Input Data for Multi Flexible Body Dynamics Analysis," NASA Tech Brief Draft, June 13, 1988. (Call author for status info 301-286-8730)

RITZ METHOD FOR TRANSIENT RESPONSE
IN SYSTEMS HAVING UNSYMMETRIC STIFFNESS

FUNDAMENTAL MODES

A structure must be defined according to its elastic distribution, its damping distribution, its mass distribution, its boundary conditions and its complement of those loading conditions which are active during its operation under a control system. Each such loading that can be applied independently of some of the other loadings should be treated as a distinct deformation-producing condition; i.e. as one producing a unique static response shape. Within the theory of linear elasticity the magnitudes of static response to a static loading varies directly as the magnitude of the loading, so a simple unit magnitude of load is all that is necessary to establish the shape of an individual mode into which the structure deforms. A sketch of a cantilevered beam with an end load illustrates the point. The loads are graduated from 1 unit, to 2 units, to 3 units.



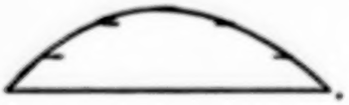
$$\frac{\delta_1}{P_1} = \frac{\delta_2}{P_2} = \frac{\delta_3}{P_3} = \frac{\delta}{P} = \frac{L^3}{3EI}$$

The ratio of deformation to load stays constant at $L^3/(3EI)$ as the load varies. The same holds true at positions other than the tip deflections. For instance, the deflection at an interior point, such as L/s where $1 < s < \infty$, is

$$\delta\left(\frac{L}{s}\right) = \frac{PL^3}{6EIs^2}\left(3 - \frac{1}{s}\right) \text{ , therefore}$$

$$\frac{\delta\left(\frac{L}{s}\right)}{P} = \frac{L^3}{6EIs^2}\left(3 - \frac{1}{s}\right) \text{ ,}$$

which shows that the ratio of deformation to load at an interior point stays constant as load varies. The notion of shape that is independent of amplitude can be easily depicted by a sketch of a violin string being played in its fundamental mode.



All three show the string sounding the same pitch (frequency) but at different loudnesses (amplitude). All 3 deformations are considered to have the same shape.

Therefore, in order to get started, a controlled structure must be exercised with a unit load for each loading condition, which can vary in magnitude independently of other loads. This will produce, what will be called, the set of fundamental modes for the complement of loading conditions. It was assumed at the start of this derivation that the set of loads produced a set of unique static modes. This assumption needs to be tested against a criterion for uniqueness. The criterion that is germain here is linear independence. The modes need to be put on an equal footing for such a test by normalizing them uniformly. There is a choice of methods for normalizing at this point. Because this is a dynamics problem, one would be inclined toward mass normalization, but for now a simple Euclidean length will suffice to put the modes on an equal footing to establish linear independence. Later, when the modes are orthogonalized, they will be normalized to mass. Normalizing to maximum or to the amplitude of a common position will be excluded, because of the bias that would be introduced in the linear independence check.

Initially static modes are to be obtained by solving n loadings.

$$(1) \qquad \left[K_{LL}\right]\left[\left\{u_1\right\},\left\{u_2\right\},\ldots\ldots,\left\{u_n\right\}\right]_L = \left[\left\{P_1\right\},\left\{P_2\right\},\ldots\ldots,\left\{P_n\right\}\right]_L$$

$$=\!=\!=\!> \quad \left[K_{LL}\right]\left[U_{Lj}\right] = \left[P_{Lj}\right].$$

Each of the $\left\{u_j\right\}$ of [U] will be individually normalized by its Euclidean length. Compute the individual normalizing constants according to

$$(2) \qquad \left[\left\{u_j\right\}^T\left\{u_j\right\}\right] = n_j.$$

Each term of $\left\{u_j\right\}$ will now be divided by the square root of the normalizing constant n_j. Name this normalized fundamental

$$(3) \qquad \text{PHI sub j, } \left\{\phi_j\right\}; \text{ i.e. } \frac{1}{\sqrt{n_j}}\left\{u_j\right\} \equiv \left\{\phi_j\right\}.$$

In order to use these static deformations as generalized coordinates, they should be linearly independent. Now we are faced with the decision as to what criterion of linear independence to use and what tolerance to allow. The classical definition of linear independence is that the Gramian > 0. The

Gramian[2] is a determinant of the matrix of the dot products of each coordinate into every other coordinate. Translated to the context in which we want to consider it, assume that the group of vectors $\phi_1 - - - - \phi_n$ is a set to be tested, then the Gramian for them is

$$(4) \qquad G(\phi_1, \phi_2, \ldots\ldots, \phi_n) = \begin{vmatrix} \phi_1^T\phi_1 & \phi_1^T\phi_2 & \phi_1^T\phi_3 \cdots\cdots & \phi_1^T\phi_n \\ \phi_2^T\phi_1 & \phi_2^T\phi_2 & \phi_2^T\phi_3 \cdots\cdots & \phi_2^T\phi_n \\ \cdots & \cdots & \cdots & \cdots \\ \phi_n^T\phi_1 & \phi_n^T\phi_2 & \phi_n^T\phi_3 \cdots\cdots & \phi_n^T\phi_n \end{vmatrix} .$$

If $G(\phi_1 \ldots\ldots \phi_n) = 0$, the set of ϕ_n are a dependent set, but if it is > 0 the set of ϕ_n are linearly independent. In a finite number system, such as that under which a digital computer operates, the establishment of a true zero is difficult, if not impossible. Consequently the decision as to what criterion to use rests on the practical consideration of how much impurity to allow in the set that we want to use as the expansion functions. If the Gramian is just slightly greater than zero, it implies that yes a functional relationship can be set up for one in the set with respect to the others, because imperfections creep in to contaminate the zero computation. Now if all ϕ's were normalized to unity and constituted a truly linearly independent set, then the value of the Gramian would be unity.

$$G(\text{unity normalized } \phi) = 1.$$

---

[2] TENSOR ANALYSIS, by I. S. Sokolnikoff; pp 6; John Wiley,1951

Any departure from 1 in the Gramian of this unity normalized set means that some imperfection is creeping in and the further the descent from 1 towards zero implies more and more imperfections away from true linear independence. So What! Our goal is to have a good set of vectors so that when we expand our solution in them, we will get good accuracy and get good convergence. If we have a set of linearly independent vectors but too few of them to span the range of actions in which our structure will operate, we will fail to converge close enough to a correct solution. If we admit too many vectors that are almost independent, but do have some imperfections, the answers will contain biased emphases and will definitely contribute amplitudes that are too great in some modes. In further consideration of the practical factors that will govern our decision, we ask, "What will it cost to find out if there is linear independence?" In the case of the Gramian, the computations involve the creation of a full matrix of vector dot products that must then be decomposed and finally the product of all diagonal terms of the decomposed matrix forms the Gramian determinant. Decompositions are expensive, so a method other than using the Gramian would be worth while to investigate.

Another approach is to look at the ingredients of the Gramian i.e. the individual matrix dot products. By definition the dot product of 2 vectors **A** and **B** is

$\mathbf{A} \cdot \mathbf{B} = |A| \times |B| \cos \theta$, where θ is the generalized angle between the 2 vectors. This can be extended to vectors in N-dimensional space. Set up a criterion based upon the size of the angle that a trial vector ϕ makes with each of those ϕ_n that have already been judged linearly independent.

BEST COPY
AVAILABLE

(5) $\theta_{k,n} = \cos^{-1}\left[\frac{\phi_n}{|\phi_n|}\right]^T\left[\frac{\phi_k}{|\phi_k|}\right] \geq \tau$, where τ is some threshhold value. Ideally τ would be $\pi/2$ for an orthogonal set with perfect linear independence. The poorest possible value would be that for which ϕ_n and ϕ_k are coincident, i.e. zero angle. This criterion can be rephrased by saying that it tests how well $\cos\theta$ compares with $\cos\pi/2$

(6) $$\left(\cos\theta_{k,n}\right) - \left(\cos\frac{\pi}{2}\right) < \kappa.$$

The desire is to hold the angle between test vectors to be somewhere between a threshhold and $\pi/2$. If the threshhold angle were $\pi/3$, $\kappa = \cos\pi/3 = .5$, then the test would require the cosine of the angle between test vectors to be less than 0.5 :

$$\left|\left[\frac{\phi_n}{|\phi_n|}\right]^T\left[\frac{\phi_k}{|\phi_k|}\right]\right| < 0.5 = \kappa \quad \text{for all } 1 < n < k.$$

Once κ has been decided upon, the test is carried out against every ϕ_n for $n < \kappa$.

In the case of the cosine test, the matrix of dot products would have to be formed as in the case of the Gramian, but no decomposition need be done. The absolute value of each term is compared to κ.

The Gramian test has the advantage of making its decision by comparing only one number against a threshold while the cosine test involves comparison of every ratio in a column against a threshhold. The term by term processing involves only simple operations; the net result is that the cosine test is much less expensive than the Gramian test. The cosine test has been chosen for this method.

HARMONICS

Modes, determined solely on the basis of static loads, are questionable to apply without supplement to the solution of dynamics, because they are devoid of inertia effects. Supplemental modal vectors can be generated by finding the deformation due to forces derived from accelerating masses distributed through the structure by an amplitude equal to the vector of elastic deformation. Call the deformation from inertia effects, accelerated by amplitudes derived from the fundamental mode, the first harmonic. A second harmonic can be generated by the scheme used to generate the first harmonic, except that the the inertias are now accelerated through amplitudes derived from the first harmonic. Similarly, a third harmonic can be generated from the acceleration of mass through the amplitudes of the second harmonic, etc. Eventually the upper harmonics will tend towards congruence, so there will be an nth harmonic beyond which no distinct modes will be added. The measure to be used for finding the useful limit will be linear independence.

The set of linearly independent modes, consisting of a group of fundamentals plus groups of harmonics associated with each fundamental, when normalized to the Euclidean length, then orthogonalized, will be used as modes of generalized coordinates in expanding the behavior of a structure under the management of a control system. The mathematics of these inertia modes follows. The mass of the structure $\left[M_{LL}\right]$ will be accelerated by an amplitude distributed spatially according to the shape of the normalized - but not orthogonalized - fundamental mode $\left\{\phi_f\right\}$ to create an inertia forcing for the first harmonic $\left\{F_{h1}\right\}$; i.e.

(7) $\left\{F_{h1}\right\} = \left[M_{LL}\right]\left\{\ddot{\phi}_f\right\}$, where

(8) $\left\{\ddot{\phi}_f\right\} = \left\{\phi_f \frac{d^2}{dt^2} \cos \omega t\right\} = \left\{\phi_f(-\omega^2 \cos \omega t)\right\}$.

The shape of the deformation through which the mass will be accelerated is established by ϕ_f. The effect of the term $(\omega^2 \cos \omega t)$ is to merely amplify the shape as a function of time. Our interest, at this stage of the derivation, is only in the shape and not the total dynamic response, therefore the forcing can be treated as a statics problem with the spatial distribution of the set of accelerations, limited to any instant of time, therefore

$\left|\ddot{\phi}_f\right| \simeq \left|\phi_f\right|$ and the resulting static force is

(9) $\left\{F_{h1}\right\} = \left[M_{LL}\right]\left\{\phi_f\right\}$.

Apply this force to the structure and solve for the response.

(10) $\left[K_{LL}\right]\left\{u_{h1}\right\} = \left\{F_{h1}\right\} = \left[M_{LL}\right]\left\{\phi_f\right\}$,

from which the response can be explicitly isolated:

(11) $\left\{u_{h1}\right\} = \left[K_{LL}\right]^{-1}\left[M_{LL}\right]\left\{\phi_f\right\}$.

The $\left\{u_{h1}\right\}$ so obtained will be normalized by the Euclidean length and will be tested for linear independence , which if accepted, will be named $\left\{\phi_{h1}\right\}$ and will augment the complement of Ritz modes. This first harmonic will have a shape of deformation sufficiently unique to make it worth while to consider it as the basis for accelerating the mass through its spatial behavior to obtain a second harmonic, similar to the way that the fundamental was used in the generation of the first harmonic. Mathematically the method of forming the second harmonic follows the pattern already established for the first harmonic. Form the forcing

$$\left\{F_{h2}\right\} = \left[M_{LL}\right]\left\{\ddot{\phi}_{h1}\right\}. \tag{12}$$

Let $\left\{\ddot{\phi}_{h1}\right\} = \left\{\phi_{h1}\right\} \frac{d^2}{dt^2}(\cos \omega t)$ and limit the value to its static amplitude $\left|\ddot{\phi}_{h1}\right| \simeq \left|\phi_{h1}\right|$, then solve for the static response $\left\{u_{h2}\right\}$ from

$$\left[K_{LL}\right]\left\{u_{h2}\right\} = \left\{F_{h2}\right\} = \left[M_{LL}\right]\left\{\phi_{h1}\right\} . \tag{13}$$

Extract $\left\{u_{h2}\right\}$ explicitly.

$$\left\{u_{h2}\right\} = \left[K_{LL}\right]^{-1}\left[M_{LL}\right]\left\{\phi_{h1}\right\} . \tag{14}$$

The $\left\{u_{h2}\right\}$ so obtained will be normalized by its Euclidean length and will be tested for linear independence, which if accepted, will be named $\left\{\phi_{h2}\right\}$ and will augment the complement of Ritz modes. A question arises as to the extent to which a candidate harmonic should be tested for linear independence. Should it be tested against every other vector established up to this point, or should the candidate harmonic be tested for linear independence only against its parent fundamental and the harmonics that are spawned from that fundamental alone? From an algorithmic standpoint the latter route is favored, because all quantities stay within an inner loop. The merit of this decision can be tested and be replaced if need be. It is known that as the recursion steps are carried out for higher harmonics, the deformations will tend toward congruence so there is a definite need for testing each new candidate harmonic against its parent and siblings. If there is no physical risk for so limiting the linear independence check to the family associated with just one fundamental, it will be opted for here.

A certain pattern starts to appear from the development of these two harmonics. The matrix product $\left[K_{LL}\right]^{-1}\left[M_{LL}\right]$ is used repeatedly. Consequently, it can be generated once and saved for recall in the generation of any level of harmonics of any fundamental. Name this matrix product

(15) $[SOLi] = \left[K^{*}_{LL}\right]^{-1}\left[M_{LL}\right]$, where i can take on the BCD

character for the primary or the adjoint mode, and * can take on either blank for primary or T (for transpose) for the adjoint. Discussion of adjoint mode generation will be taken up subsequently.

Capitalizing on the pattern that has been revealed, all first harmonics for all fundamentals can be generated in a series of matrix operation as follows, as adapted for the primaries:

(15A)
$$\left[\{u_1\},\{u_2\},\{u_3\},\cdots\cdot,\{u_i\},\cdots,\{u_n\}\right]_{Sh1} = \left[K_{LL}\right]_S^{-1}\left[M_{LL}\right]\left[\{\phi_1\},\{\phi_2\},\{\phi_3\},\cdots\cdot,\{\phi_i\},\cdots,\{\phi_n\}\right]_{Sf}$$

This can be compressed into

(15B)
$$\left[U_{H1}\right]_S = \left[K_{LL}\right]_S^{-1}\left[M_{LL}\right]\left[\phi_F\right]_S$$

Equations (15,15A,15B) represent all possible inertia response raw data for forming first harmonic modes of primary fundamentals.

Normalization of these responses can also be performed on all vectors treated as a matrix. The Euclidean length can be extracted from the multiplication of $\left[U_{H1}\right]$ by itself.

(15C)
$$\left[U_{H1}\right]^T\left[U_{H1}\right] = \left[U_{H1}SQ\right].$$

Strip off the diagonal, take the inverse of each, followed by its square root, to form a diagonal matrix of scale factors mode by mode.

(15D)
$$\left[\,U_{H1}SQ\,\right]^{-.5} \text{======>} \left[SCAL_{H1}\right].$$

Apply this matrix of scale factors to the first harmonic responses, of fundamental inertia loadings, as a post multiplication operation to get a set of candidate normalized harmonic modes.

(15E)
$$\left[U_{H1}\right]\left[SCAL_{H1}\right] = \left[\phi_{H1}^{CA}\right].$$

This matrix of candidate first harmonics needs to be tested for linear independence. Testing will proceed in two parts. All first harmonics will be tested against all other first harmonics, then all first harmonics will be tested against all fundamental modes. Start with the harmonics by themselves. The cosine test involves taking the dot product of every mode against every other mode. This is done in a single matrix operation.

$$(15F) \quad \left[\phi_{H1}^{CA}\right]^T \left[\phi_{H1}^{CA}\right] = \left[DOT_{H1}\right].$$

Examining the $\left[DOT_{H1}\right]$ matrix closely, one can recognize that the first row represents the dot product of the harmonic (first in sequence) against each of those in the set of harmonics. The second row represents the dot product of the harmonic (second in sequence) against each of the set of harmonics. Et cetera. Consequently, the next step is to strip off one row to examine how well this candidate harmonic holds up in the cosine test versus other first harmonics. Next another row is stripped off and this candidate is tested and so on until all candidates have been examined. If the vectors had not been initially normalized to length, it would have been necessary to do so at this point to form the cosines. As a consequence, the matrix DOT_{H1} consists of all cosine terms. Going back now to the first row, some detail will reveal a pattern for systematizing all of the candidates.

Select one term at a time starting with the 2nd and take its absolute value then compare that value with κ. Shift the index to the 3rd and do the same. Continue until either a value greater than κ is encountered or until the the end of the row is reached. If any term tests greater than κ, catalog the row number and proceed to the next row. All successful candidates will be eligible to be tested against the matrix of fundamental modes. The harmonic vs. fundamental test will patterned after the harmonic vs. harmonic test. All successful candidate harmonics will be held in reserve to form the basis of 2nd harmonics before they will be merged with, but in sequence after, the fundamental modes.

The number of successful first harmonic modes may be fewer than those in the set of fundamentals. This does not matter, because the method of computing harmonics is independent of the size of the order of the vectors from which they are derived. The generation of second harmonic modes and higher will proceed along the pattern just outlined for first harmonic modes, except that after the responses to inertia loads have been computed, the successful modes from which they were derived, will be merged into the matrix of previous Ritz modes. Now the $i^{\underline{th}}$ set of harmonics will be tested for linear independence vs. not only themselves but against all other Ritz vectors including fundamentals and all previous order successful harmonics up through the $(i-1)^{\underline{th}}$.

The generation of higher harmonics will be subject to a choice of two limitations. The analyst may want to limit the maximum number of harmonics to be admitted for any particular investigation because of, say, a study in a low frequency domain. He can invoke such control by giving the value of the maximum number of harmonics to the parameter MODSPEC. Declining to assign a value to MODSPEC will cause the number of harmonics to be limited by those that pass the linear independence check.

ADJOINT MODES

For the non-symmetric problem, adjoint vectors are required to obtain the reduced order coefficient matrices. There are no set rules for introducing the adjoint basis. In the spirit of Lanczos method, a trial method is introduced and refined. For lack of any better trial scheme, the starting matrices of the adjoint system that will be used here will consist of the transpose of the original $\left[K_{LL}\right]$ matrix; i.e. $\left[K_{LL}^T\right]$, (which will also be non-symmetric), and the original load vectors $\left\{P_L\right\}$.

The static solution of this adjoint system under the n original loads yields a set of responses $\left\{v_j\right\}$:

(16) $$\left[K_{LL}^T\right]\left[\left\{v_1\right\},\left\{v_2\right\},\ldots\ldots,\left\{v_n\right\}\right] = \left[\left\{P_1\right\},\left\{P_2\right\},\ldots\ldots,\left\{P_n\right\}\right]$$

$$====> \quad \left[K_{LL}^T\right]\left[V_{Lj}\right] = \left[P_{Lj}\right] .$$

Each of the $\left\{v_j\right\}$ of [V] will be individually normalized by its Euclidean length. Compute the individual normalizing constants according to

(17) $$\left[\left\{v_j\right\}^T\left\{v_j\right\}\right] = \alpha_j .$$

Each term of $\left\{v_j\right\}$ will now be divided by the square root of the normalizing constant α_j. Name this normalized fundamental

(18) PSI sub j, $\left\{\psi_j\right\}$; i.e. $\frac{1}{\sqrt{\alpha_j}}\left\{v_j\right\} \equiv \left\{\psi_j\right\}$.

Each adjoint fundamental will be checked for linear independence against all other adjoint fundamentals. If any adjoint fundamental mode fails the linear independence check, and if its primary companion passed, both primary and adjoint will be discarded. This is necessary in order to retain a uniform sequence when setting up generalized mass and generalized stiffness. The cosine test will be used to certify linear independence. Inertial harmonics of adjoint fundamentals will be generated in the same manner as those of the primary fundamentals. Once again the linear independence of the adjoint harmonics will be checked only against its parent and siblings, instead of the currently established set of **ALL** Ritz modes.

ORTHOGONALIZATION

The solution of the differential equations is enhanced if the generalized coordinates used to span the response space are orthogonal. Our set of linearly independent vectors can be orthogonalized. At this point we have a pair of bases vectors $\{\phi\}$ and $\{\psi\}$ that are each separately linearly independent. After orthogonalizing they will be given the symbols $\{\zeta\}$ and $\{\Omega\}$ respectively. Options for orthogonalization and for associated constraint weightings were given due consideration. Before deciding on what options to choose, it will help to review the ultimate application.

The dynamic equation in metric coordinates is:

$$(19) \qquad [M]\{\ddot{x}\} + [B]\{\dot{x}\} + [K]\{x\} = F(t).$$

Transform the metric coordinates to the primary bases vectors as generalized coordinates. Assume that the orthogonalization of ϕ into ζ has already taken place. Let

(20) $\{x\} = [\zeta]\{\xi\}$, giving

$$[M][\zeta]\{\ddot{\xi}\} + [B][\zeta]\{\dot{\xi}\} + [K][\zeta]\{\xi\} = F(t).$$

At this point the columns only of the coefficient matrices M, B, and K have been transformed to generalized form. Next the rows of the matrices must be transformed. In the symmetric case[3] this is done by using the transform of the symmetric modes. However, in this the unsymmetric case, we pre-multiply by the transpose of the adjoint bases vectors. Assume that the orthogonalization of ψ into Ω has taken place.

(22) $$[\Omega]^T[M][\zeta]\{\ddot{\xi}\} + [\Omega]^T[B][\zeta]\{\dot{\xi}\} + [\Omega]^T[K][\zeta]\{\xi\} = [\Omega]^T F(t).$$

The desire is to decouple the equations as much as possible. To start with, we want the generalized mass to be a unity matrix; i.e.

(23) $$[\Omega]^T[M][\zeta] \equiv \left[{}^{\diagdown} I_{\diagdown}\right].$$

Constraining the generalized mass to unity will affect the normalization to such an extent that the generalized coordinates will have been transformed to final form. Thus, much of the control in obtaining diagonalized damping and diagonalized stiffness will have been removed, so it appears that the generalized stiffness matrix may be coupled. Since $\left[{}^{\backslash}I_{\backslash}\right]$ is a square matrix, the requirement of equation (23) implies that the order of the adjoint vectors Ω be the same as the order of the primary vectors ζ.

---

[3] Dynamics of Structures, by W. C. Hurty & M. F. Rubinstein, pp 125, Prentice Hall, 1964

A. Use Gram-Schmidt method for orthogonalizing the Primary Ritz modes and apply the simple constraint of unit diagonalizing with no weighting.

B. But in the case of Adjoint Ritz modes the orthogonalized set will be expanded in terms of the complete set of adjoint bases with the dual constraint of equation (23) having mass weighting.

Mathematically these statements translate into building the normalized vectors as follows.

A. Self-Orthogonalization

To start with, the simple Gram-Schmidt method sets up a matrix of undetermined coefficients to involve an increasing number of bases vectors ϕ in the content of the orthogonalized vectors ζ.

$$\begin{aligned}
&\{\zeta_1\} = \{\phi_1\} \\
&\{\zeta_2\} = a_{11}\{\phi_1\} + \{\phi_2\} \\
&\{\zeta_3\} = a_{21}\{\phi_1\} + a_{22}\{\phi_2\} + \{\phi_3\} \\
&\vdots \\
&\{\zeta_{i+1}\} = a_{i1}\{\phi_1\} + a_{i2}\{\phi_2\} + \cdots + a_{ii}\{\phi_i\} + \{\phi_{i+1}\} \\
&\vdots \\
&\{\zeta_n\} = a_{n-1,1}\{\phi_1\} + a_{n-1,2}\{\phi_2\} + \cdots + a_{n-1,n-1}\{\phi_{n-1}\} + \{\phi_n\}.
\end{aligned} \tag{24}$$

BEST COPY
AVAILABLE

The self normalizing constraint is written in matrix notation as:

(25) $$\left[\zeta_{Li}\right]^T\left[\zeta_{Lj}\right] = \left[{}^{\backslash}\delta_{ij\backslash}\right]$$

One can better appreciate the earlier topic of having to decide on sequencing during generation of ingredients of the bases in light of the character of orthogonalization. If the fundamentals are sequenced together at first, then the initial orthogonalized vectors will contain a minimum of higher harmonics in their expansion. We ask, Is this good or bad? If inertia effects dominate the dynamic behavior of a structure under certain loads, probably sequencing harmonics in earlier might help. But in this study the option was taken to group the fundamentals ahead of the harmonics instead of layering one fundamental and all of its higher harmonics on top of a second fundamental and all of its higher harmonics et cetera.

Trace the effect of the orthogonality constraint on achieving a solution for the undetermined coefficients. Operate on the first two equations of the set in equation (24).

(26) $$\left\{\zeta_1\right\} = \left\{\phi_1\right\}. \qquad \left\{\zeta_2\right\} = a_{11}\left\{\phi_1\right\} + \left\{\phi_2\right\}.$$ Impose the single orthogonality constraint of equation (25).

(27) $$\left\{\zeta_1\right\}^T\left\{\zeta_2\right\} = 0 \xRightarrow{\text{Subs for } \zeta_2} \left\{\zeta_1\right\}^T\left\{a_{11}\left\{\phi_1\right\} + \left\{\phi_2\right\}\right\} = 0 \xRightarrow{\text{Multiply}}$$

(28) $$\left\{\zeta_1\right\}^T\left\{\phi_1\right\}a_{11} + \left\{\zeta_1\right\}^T\left\{\phi_2\right\} = 0. \qquad a_{11} = (-)\,\frac{\left\{\zeta_1\right\}^T\left\{\phi_2\right\}}{\left\{\zeta_1\right\}^T\left\{\phi_1\right\}}\,.$$

Substitute from (26) $a_{11} = (-)\dfrac{\{\Phi_1\}^T\{\Phi_2\}}{\{\Phi_1\}^T\{\Phi_1\}}$. The coefficient a_{11} is now expressed entirely in terms of the set of known normalized bases vectors $[\Phi]$. Substitute a_{11} into the equation above and now $\{\zeta_2\}$ is known. Turn to the third vector.

(29) $\{\zeta_3\} = a_{21}\{\Phi_1\} + a_{22}\{\Phi_2\} + \{\Phi_3\}$. Two orthogonality conditions are imposed between ζ_3 and the previously found ζ_1 and ζ_2.

$$\begin{aligned} \{\zeta_1\}^T\{\zeta_3\} = 0 \qquad & \{\zeta_1\}^T\left\{a_{21}\{\Phi_1\} + a_{22}\{\Phi_2\} + \{\Phi_3\}\right\} = 0 \\ \xRightarrow{\text{Subs for } \zeta_3} \qquad & \\ \{\zeta_2\}^T\{\zeta_3\} = 0 \qquad & \{\zeta_2\}^T\left\{a_{21}\{\Phi_1\} + a_{22}\{\Phi_2\} + \{\Phi_3\}\right\} = 0 \end{aligned} \tag{30}$$

$$\begin{aligned} \{\zeta_1\}^T\{\Phi_1\}a_{21} + \{\zeta_1\}^T\{\Phi_2\}a_{22} &= (-)\{\zeta_1\}^T\{\Phi_3\} \\ \{\zeta_2\}^T\{\Phi_1\}a_{21} + \{\zeta_2\}^T\{\Phi_2\}a_{22} &= (-)\{\zeta_2\}^T\{\Phi_3\} \end{aligned} \tag{31}$$

This gives two algebraic equations in the two unknowns a_{21} and a_{22}.
Substituting these coefficients makes $\{\zeta_3\}$ known. Turn to the $i+1^{\underline{th}}$ vector.

BEST COPY
AVAILABLE

$$(32)\qquad \{\zeta_{i+1}\} = a_{1i}\{\phi_1\} + a_{2i}\{\phi_2\} + \ldots\ldots + a_{ii}\{\phi_i\} + \{\phi_{i+1}\} =$$

$$= \left[\{\phi_1\},\{\phi_2\},\ldots\ldots\{\phi_i\},\{\phi_{i+1}\}\right]\begin{Bmatrix} a_{1i} \\ a_{2i} \\ \cdot \\ a_{qi} \\ \cdot \\ a_{ii} \\ 1 \end{Bmatrix} = [\phi]_{i+1} \lfloor\, a \,\rfloor^T_{i+1}$$

Assemble the i constraint conditions in preparation for substituting ζ_{i+1} from equation (32) into them.

$$(33)\qquad \{\zeta_1\}^T\{\zeta_{i+1}\} = \{\zeta_2\}^T\{\zeta_{i+1}\} = \ldots = \{\zeta_i\}^T\{\zeta_{i+1}\} = 0,$$

which can be combined into

$$(33A)\qquad \left[\{\zeta_1\}^T_{,}\{\zeta_2\}^T_{,}\{\zeta_3\}^T_{,}\ldots\{\zeta_i\}\right]\ \{\zeta_{i+1}\} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}.$$

Now substitute the expansion in terms of $\{\phi\}$.

$$(34)\qquad \left[\{\zeta_1\}^T,\{\zeta_2\}^T,\ldots\ldots,\{\zeta_i\}^T\right]\ \left[\{\phi_1\},\{\phi_2\},\ldots\ldots\{\phi_i\},\{\phi_{i+1}\}\right]\begin{Bmatrix} a_{1i} \\ a_{2i} \\ \cdot \\ a_{qi} \\ \cdot \\ a_{ii} \\ 1 \end{Bmatrix} = 0.$$

Which can be compressed into

$$(35)\qquad [\zeta]^T_i[\phi]_{i+1}\lfloor\, a \,\rfloor^T_{i+1} = 0.$$

But by introducing partitions into $[\phi]$ and $\{a\}$ it can be written more intuitively as

$$(36)\quad [\zeta]^T_i\left[[\phi]_i \,\vdots\, \{\phi\}_{i+1}\right]\begin{Bmatrix} a_1 \\ \cdot\,\vdots \\ 1 \end{Bmatrix} = 0. \qquad ===> \left[[\zeta]^T_i[\phi]_i \,\vdots\, [\zeta]^T_i\{\phi\}_{i+1}\right]\begin{Bmatrix} a_1 \\ \cdot\,\vdots \\ 1 \end{Bmatrix} = 0.$$

$$===> [\zeta]^T_i[\phi]_i\{a\}_i + [\zeta]^T_i\{\phi\}_{i+1} = 0. \quad ===> [\zeta]^T_i[\phi]_i\{a\}_i = (-)[\zeta]^T_i\{\phi\}_{i+1}$$

from which all of the undetermined coefficients $a_i, a_2, \ldots, a_i$ can be computed and substituted into equation (32) to evaluate ζ_{i+1}.

For solutions of succeeding vectors i+2, i+3, ...n; the square coefficient matrix $[\zeta]_i^T[\phi]_i$ increases incrementally in order up to n x n, so that results from the previous calculation might be considered for saving. Then the increments can be merged into the salvaged core to continue on. Details of the strategy will be given in the CODING document.

B. Dual Orthogonalization

The dual orthogonalization of the adjoint bases are organized into a full matrix of undetermined coefficients for the expansion of normalized vectors Ω as components of the raw adjoint bases ψ.

$$(37) \quad \begin{Bmatrix} \{\Omega_{L1}\} \\ \{\Omega_{L2}\} \\ \{\Omega_{L3}\} \\ \vdots \\ \{\Omega_{Ln}\} \end{Bmatrix} = \begin{pmatrix} b_{11} & b_{12} & b_{13} & \cdots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \cdots & b_{2n} \\ b_{31} & b_{32} & b_{33} & \cdots & b_{3n} \\ \vdots & & & & \vdots \\ b_{n1} & b_{n2} & b_{n3} & \cdots & b_{nn} \end{pmatrix} \begin{Bmatrix} \{\psi_{L1}\} \\ \{\psi_{L2}\} \\ \{\psi_{L3}\} \\ \vdots \\ \{\psi_{Ln}\} \end{Bmatrix} .$$

This can be written in more conventional form as

$$(38) \quad \left[\{\Omega_{L1}\}, \{\Omega_{L2}\}, \cdots \{\Omega_{Ln}\}\right] = \left[\{\psi_{L1}\}, \{\psi_{L2}\}, \cdots \{\psi_{Ln}\}\right] \begin{pmatrix} b_{11} & b_{21} & \cdots & b_{n1} \\ b_{12} & b_{22} & \cdots & b_{n2} \\ b_{13} & b_{23} & \cdots & b_{n3} \\ \vdots & & & \vdots \\ b_{1n} & b_{2n} & \cdots & b_{nn} \end{pmatrix} .$$

Note the the b_{ij}'s in equation (38) are the transpose of the b_{ji}'s in equation (37). Equation (38) can be condensed to

(39) $\left[\Omega_{Ln}\right] = \left[\psi_{Lp}\right]\left[\beta_{pn}\right]$, in which [β] is the transpose of the b_{ij}'s.

The mass orthogonality constraint, according to the transformation requirements of the dynamic differential equation (23), is a relationship between the normalized adjoint and the normalized primary bases; i.e. $\Omega^T_{Li}M_{LL}\zeta_{Lj} = \delta_{ij}$. Since the M and ζ are known, their transposes can be taken immediately, so it becomes better strategy to transpose the constraint equation in order to obtain Ω in non-transposed form; i.e.

(40) $$\left[\zeta_{Lj}\right]^T\left[M_{LL}\right]\left[\Omega_{Li}\right] = \left[{}^{\setminus}\delta_{ji}{}_{\setminus}\right].$$

Now substitute from the expansion equation (39) into the mass orthogonality constraint equation (40). Confine to one index at a time. Set i = 1.

(41) $$\left[\zeta_{Lj}\right]^T\left[M_{LL}\right]\left\{\Omega_{L1}\right\} = \left[\zeta_{Lj}\right]^T\left[M_{LL}\right]\left[\psi_{Lp}\right]\left\{\beta_{p1}\right\} = \delta_{j1} = \begin{cases} 1 & \text{for } j = 1 \\ 0 & \text{for } j \neq 1 \end{cases}$$

This will produce a solution to the first column of β_{p1}. Next set i = 2 and substitute for Ω_{L2} from equation (39).

(42) $$\left[\zeta_{Lj}\right]^T\left[M_{LL}\right]\left\{\Omega_{L2}\right\} = \left[\zeta_{Lj}\right]^T\left[M_{LL}\right]\left[\psi_{Lp}\right]\left\{\beta_{p2}\right\} = \delta_{j2} = \begin{cases} 1 & \text{for } j = 2 \\ 0 & \text{for } j \neq 2 \end{cases}$$

This will produce a solution to the second column of β. A pattern is now apparent for solving for the complete content of β in a single operation, by recognizing the coefficient $\left[\zeta_{Lj}\right]^T\left[M_{LL}\right]\left[\psi_{Lp}\right]$ is the same in every equation; only the columns of unknown β's and the columns of the δ_{ji} change. Combine the columns into matrices; i.e.

(43) $$\left[\zeta_{Lj}\right]^T\left[M_{LL}\right]\left[\psi_{Lp}\right]\left[\beta_{pi}\right] = \left[{}^{\setminus}I_{\setminus}\right].$$

Isolate β_{pi}.

$$(44) \qquad \left[\beta_{pi}\right] = \left[{}^{\backslash}I_{\backslash}\right]\left[\zeta^T_{Lj} M_{LL} \psi_{Lp}\right]^{-1}.$$

Substitute into the defining equation for normalized adjoints (39).

$$(45) \qquad \left[\Omega_{Ln}\right] = \left[\psi_{pn}\right]\left[\beta_{pn}\right] = \left[\psi_{Lp}\right]\left[{}^{\backslash}I_{\backslash}\right]\left[\zeta^T_{Lj} M_{LL} \psi_{Lp}\right]^{-1}.$$

All quantities are now derived for getting a solution of the dynamic differential equation by using Ritz modal vectors. One thing not taken up however, was the definition of damping so as to give as sparse a generalized damping matrix as possible. The other topic that still needs addressing is data recovery. These topics will be reserved for an extension to the basics as developed here. Details of converting this theory to DMAP coding has been published in a report entitled "RITZ MODES FOR UNSYMMETRIC MATRICES--DMAP CODING OF THE THEORY" by Thomas B. Butler.

The coding was done in 3 steps. (1) Fundamental Primary and Adjoint modes were obtained from a DMAP ALTER to the Statics Rigid Format. The listing of this code is attached as Appendix A. (2) Harmonics for the Primary and Adjoint sets were coded as a pure DMAP approach. The listing of this code is attached as Appendix B. (3) Orthogonalization was coded as a pure DMAP approach. The listing of this code is attached as Appendix C.

A simple demonstration problem was used to certify the method and the coding. It was run twice with a different threshhold value of Kappa to exercise a number of different

BEST COPY
AVAILABLE

paths. Details of the demonstration problem are given in Appendix D. Generalized Mass and Generalized Stiffness that were produced in the two different runs are given in Appendix E. One run set the linear independence threshhold Kappa to 0.007. Only the four fundamentals passed the linear independence test, so the generalized mass and stiffness are only of order 4 x 4. Kappa was deliberately set high to a value of 0.95 in the second run so as to admit 5 harmonics in addition to the four fundamentals. Resulting generalized mass and stiffnesses are of order 9 x 9.

The generalized mass in both cases was practically unity. Off diagonal terms were at least 14 orders of magnitude less than those on the diagonal. Marked differences show up in the generalized stiffneses for the two cases. When a mathematically logical value of Kappa is used as in the .007 case, the terms on the diagonal dominate the off-diagonal terms, implying very weak coupling between Ritz modes. This weak coupling could very well justify the use diagonal matrices and so benefit from a decoupled solution. When an improbable value of Kappa is used as in the 0.95 case, off-diagonal terms are large. This observation can lead to a manageable criterion for completeness of the modes.

RITZ METHOD FOR TRANSIENT RESPONSE IN SYSTEMS HAVING UNSYMMETRIC STIFFNESS

CONCLUSIONS

The method is workable. DMAP coding has been automated to such an extent by using the device of bubble vectors,[4] that it is useable for analyses in its present form. This feasibility study demonstrates that the Ritz Method is so compelling as to warrant coding its modules in FORTRAN and organizing the resulting coding into a new Rigid Format.

Even though this Ritz technique was developed for unsymmetric stiffness matrices, it offers advantages to problems with symmetric stiffnesses. If used for the symmetric case the solution would be simplified to one set of modes, because the adjoint would be the same as the primary. Its advantage in either type of symmetry over a classical eigenvalue modal expansion is that information density per Ritz mode is far richer than per eigenvalue mode; thus far fewer modes would be needed for the same accuracy and every mode would actively participate in the response. Considerable economy can be realized in adapting Ritz vectors for modal solutions. This new Ritz capability now makes NASTRAN even more powerful than before.

---

[4] Butler, T. G. and Famidi, F. R. "Bubble Vector in Automatic Merging", Proceedings of the Fifteenth NASTRAN Colloquium, 1987

BEST COPY
AVAILABLE

APPENDIX A

DMAP CODING FOR
PRIMARY AND ADJOINT FUNDAMENTALS

```
$$ ADVISABLE TO INCLUDE THIS NASTRAN CARD IN JOBS.
$
$    NASTRAN  MAXFILES = 60, FILES = (INPT,INP1,INP2,INP3,INP4)
$ RITZFUND.DMP $ DMAP ALTER FOR GENERATING FUNDAMENTAL RITZ VECTORS $
ALTER  2 $  ALTERS FOR 1988 VERSION OF NASTRAN
PARAML       CONTK/ /*PRESENCE*/ / / /V,N,NOCONTK $ CONTK IS A DMAP INPUT $
$$                  $ NOCONTK = -1 IF CONTK IS MISSING  $
PRTPARM  / /0/C,N,NOCONTK $DB
COND         ERROR4,NOCONTK $ ABORTS IF USER OMITS CONTK.
$$  AFTER GP3
ALTER 26,26  $  REPLACES PARAM STATEMENT WITH ONE THAT ENABLES MASS GENERATION
PARAM        //*ADD*/NOMGG/1/0 $ ALERTS EMG TO GENERATE MGG $
ALTER  39 $ AFTER EMA OUTPUTS KGGX FOR STIFFNESS
ADD          CONTK,KGGX/NSKGG/ $ THIS IS THE NON-SYMMETRIC STIFFNESS.
ALTER 45  $ AFTER EMA OUTPUTS MGG
PURGE    MNN,MFF,MAA/NOMGG $
ALTER    61,61 $ DON'T PURGE QG.
PURGE    KRR,KLR,QR,DM/REACT/GM/MPCF1/GO,KOO,LOO,PO,OUUV/OMIT/PS,
         KFS,KSS/SINGLE $
ALTER    62,62 $ ADD MGG TO EQUIV
EQUIV    KGG,KNN/MPCF1/MGG,MNN/MPCF1 $
ALTER    65,69 $ REPLACE MCE2 & SCE1 WITH NON-SYM OPN'S
VEC      USET/GNVEC/*G*/*M*/*N* $ 1'S ON N
PARTN    KGG,GNVEC, /KMM,KNM,KMN,KNNBAR/-1 $ -1 MEANS THAT GNVEC IS USED
$$  FOR BOTH ROW AND COL PARTNG, BUT DOES NOT MEAN THAT KGG IS SYMMETRIC
MPYAD    KNM,GM,KNNBAR/KNN1/0/+1/+1 $
MPYAD    GM,KMN,KNN1/KNN2/+1/+1/+1 $
```

RITZ METHOD FOR TRANSIENT RESPONSE IN SYSTEMS HAVING UNSYMMETRIC STIFFNESS

```
MPY3      GM.KMM.KNN2/KNN/0 $
MCE2      USET.GM, ,MGG, , /KWASTE.MNN, ,  $
LABEL     LBL2 $
EQUIV     KNN.KFF/SINGLE/MNN.MFF/SINGLE $
COND      LBL3.SINGLE $
VEC       USET/NFVEC/*N*/*F*/*S* $ 1'S ON S
PARTN     KNN.NFVEC, /KFF.KSF.KFS.KSS/-1 $
SCE1      USET, ,MNN, , / , , ,MFF, , $
ALTER     71,71 $ ADD MFF TO EQUIV
EQUIV     KFF.KAA/OMIT/MFF.MAA/OMIT $
ALTER  73,82 $ REPLACE SMP1.RBMG1.RBMG2.RBMG3 WITH UNSYM OPN'S
VEC       USET/FAVEC/*F*/*A*/*O* $
PARTN     KFF.FAVEC, /KAABAR.KOA.KAO.KOO/-1 $
DECOMP    KOO/KOL.KOU/0/0/3.N.KOMIN/S.N.KODET/S.N.KOPWR/S.N.DOSING $ UNSYM
$$  DECOMP OF KOO
FBS       KOL.KOU.KOA/GO/0/-1 $
MPYAD     KAO.GO.KAABAR/KAA/0/+1/+1 $
SMP2      USET.GO.MFF/MAA $
LABEL     LBL5 $
EQUIV     KAA.KLL/REACT/MAA.MLL/REACT $
COND      LBL6.REACT $
VEC       USET/ALVEC/*A*/*L*/*R* $
PARTN     KAA.ALVEC, /KLL.KRL.KLR.KRR/-1 $
RBMG1     USET, .MAA/ , , .MLL.MLR.MRR $
LABEL     LBL6 $
OUTPUT1, ,,,,/ /-1/4 $ SET THE DEFAULT LABEL
OUTPUT1   MLL.KLL., ,/ /0 /4 $ FOR ORTHOGONALIZATION AND FINAL CHECK
DECOMP    KLL/LLL.ULL/0/0/S.N.KLMIN/S.N.KLDET/S.N.KLPWR/S.N.KLSING $
COND      LBL7.REACT $
FBS       LLL.ULL.KLR/DM/0/-1 $
MPYAD     KRL.DM.KRR/X/0/+1/+1 $
DECOMP    X/XL.XU/0/0/S.N.XMIN/S.N.XDET/S.N.XPWR/S.N.XSING $
```

RITZ METHOD FOR TRANSIENT RESPONSE IN SYSTEMS HAVING UNSYMMETRIC STIFFNESS

```
DECOMP   KRR/KRLL.KRUU/0/0/S.N.KRMIN/S.N.KRDET/ S.N.KRPWR/S.N.KRSING $
PARAMR   //C.N.DIV/ / / /V.N.RBEPS/V.N.XDET/V.N.KRDET $
PRTPARM  //0/C.N.RBEPS $$ RIGID BODY TRANSFORMATION CHECK
LABEL    LBL7 $
$$USER INPUT = CONTK. A matrix of control properties of the same order as KLL.
$$    = SCVEC. A vector of a leading 1 followed by zeroes of order equal to
$$      the number of loading subcases.
$$    = SCADJ. A vector of leading zeroes followed by a trailing 1 of order
$$      equal to the number of loading subcases.
$$    = KAPPA. A real double precision parameter for setting the tolerance of
$$      the cosine linear independence check.
$$    = MODSPEC. A parameter for setting the integer maximum of harmonics
$$      to be generated.
$$    = LONGONE. A Vector of ones whose length is greater than twice the
$$      number of subcases times MODSPEC plus one. L= 2(SC)(1 + MODSPEC)
$+++++++++++++++++++++++++++++++++++++
FBS      LLL.ULL.MLL/SOLP  $ INERTIA COEFFICIENT FOR HARMONIC RITZ P VECTORS
$$                           PER EQUATION (15)
$  INTRODUCE THE SOL'N OF THE TRANSPOSE OF KLL TO DEVELOP INVERSE VECTORS
TRNSP    KLL/KLLT/ $
DECOMP   KLLT/TLLL.TULL/0/0 $
FBS      TLLL.TULL.MLL/SOLA $ INERTIA COEFFICIENT FOR HARMONIC RITZ A VECTORS
$$                            PER EQUATION (15)
$++++++++++++++++++++++++++++++++++++++++
ALTER 88.92 $ REPLACE SSG3 WITH NON SYM OPN'S
FBS      LLL.ULL.PL/ULV/ $
FBS      TLLL.TULL. PL /TULV/ $PAY THE PRICE OF DOING FBS ON INVERSE TOO
COND     NOUOOV.OMIT $
FBS      KOL.KOU.PO/UOOV/ $
LABEL    NOUOOV $
```

BEST COPY AVAILABLE

```
$
$$  CALCULATION OF RULV IN CONNECTION WITH IRES WILL BE IGNORED BECUZ IT WOULD
$$  BE ADVISABLE TO DO THIS CHECKING WITH SYMMETRIC MATRICES.
$ ======> USER SHOULD ESCHEW USE OF IRES <=======
$ THE EPSILON SUB E CHECK WILL BE DONE IN DBL PREC BECUZ PARAMD IS NOW AVAILABLE
$
MPYAD    KLL,ULV,PL/DELPL/0/-1/+1 $
MPYAD    ULV,DELPL, /DELWORK/+1/+1 $
MPYAD    PL,ULV, /ALLWORK/+1/+1 $
SCALAR   DELWORK/ /1/1/ /V.N.EPSNUM $
SCALAR   ALLWORK/ /1/1/ /V.N.EPSDEN $
PARAMD   / /*DIV*/V.N.EPSUBE/V.N.EPSNUM/V.N.EPSDEN $
PRTPARM  //0/C.N.EPSUBE $$ RIGID BODY TRANSFORMATION CHECK
PARAM    / /*ADD*/V.N.ADJCYC/+1/0 $ VALUE OF PARAM REMAINS POSITIVE DURING
$$                                         PROCESSING OF PRIMARIES.
COPY     LONGONE/CLONONE/ 0  $
ADD      LONGONE,CLONONE/LONGNULL/(-1.0,0.0) $
PARAML   SCVEC/ /*TRAILER*/2/V.N.VECRO $ ROW SIZE IS READ FROM SCVEC
PARAML   ULV/ /*TRAILER*/1/V.N.ZCOL $ COL SIZE IS READ FROM ULV
PARAM    / /*EQ*/V.N.LODNO/V.N.VECRO/V.N.ZCOL $ LODNO IS NEGATIVE IF VECRO=ZCOL
COND     ADJLUP,LODNO $ CONTINUE IF SCVEC AND ULV AGREE
JUMP     ERROR4 $ ABORT IF SCVEC AND ULV DON'T AGREE
FILE     FALCCLI= SAVE/FALRRLI= SAVE $
LABEL    ADJLUP $ TOP OF LOOP PRIOR TO FORMATION OF EITHER PRIM OR ADJ FUND
COND     ADJTRN,ADJCYC $ ADJCYC INITIALLY IS POS TO GIVE PRIORITY TO PRIMARY
COPY     ULV/FLV/ 0 $ FLV WILL REMAIN AN INTERNAL DATA BLOCK TO THIS FUND LOOP
EQUIV    ULV,CLONFLV $ CLONFLV IS INTERNAL. EQUIV WILL BE BROKEN AT TOP'O LOOP
JUMP     PRIMSEG $ GO AROUND THE AJOINT PREP
LABEL    ADJTRN $
COPY     TULV/FLV/ 0 $ FLV IS INTERNAL
EQUIV    TULV,CLONFLV $ CLONFLV IS INTERNAL
LABEL    PRIMSEG $
```

BEST COPY
AVAILABLE

```
MPYAD    CLONFLV.FLV. /FLSQ/+1 $
DIAGONAL FLSQ/SCALF/*DIAGONAL*/C.N.-0.5 $
MPYAD    FLV.SCALF. /FMOD/ 0 $ CANDIDATE MODE NORMALIZED TO EUCLID LGTH.
COPY     FMOD/FCLON/ 0 $
MPYAD    FCLON.FMOD. /FDOT/+1 $ MATRIX ORDERS = ZCOL.
MATPRN   FDOT.,,,// $DB
COPY     SCVEC/SCVECI/ 0 $ ROW BUBBLE STARTING FROM HEAD
COPY     SCVEC/LIVECI/ 0 $ COL BUBBLE STARTING FROM HEAD
COPY     SCVEC/MODPARTN/ 0 $ DUMMY TO BE USED FOR SWITCHING WITHIN LOOPS
ADD      LIVECI.SCVEC/FALCCLI/ /(-1.0.0.0) $ NULL BUT SAME LENGTH AS SCVEC
COPY     FALCCLI/FALRRLI/ -1 $ NULL SAME LGTH AS SCVEC
PARAM    / /*MPY*/V.N.ROCNT/ 1 / 0 $ RESET ROW COUNT TO 0 BEFORE LI CHECK
LABEL    LIRLUP $ TOP OF ROW PORTION OF LINEAR INDEPENDENCE LOOP
PARAM    / /*MPY*/V.N.NORFAL/+1/-1 $ SET DEFAULT TO NEG TO JUMP OVER FAIL BOOK
PARTN    FDOT. .SCVECI/ .ROCAI. . /+7/+2 $
PARAM    / /*ADD*/V.N.ROCNT/V.N.ROCNT/ 1 $ ROW COUNT MONITOR INCREMENTED BY ONE
PARAM    / /*MPY*/V.N.CLCNT/ 1 /V.N.ROCNT $SET COL COUNT=ROW COUNT PRIOR LI CHK
LABEL    LICLUP $ TOP OF COLUMN PORTION OF LINEAR INDEPENDENCE LOOP
PARAM    / /*ADD*/V.N.CLCNT/V.N.CLCNT/ 1 $ COL COUNT MONITOR INCREMENTED BY ONE
SCALAR   ROCAI/ /1/V.N.CLCNT/ /V.N.RCF $ COSINE TERM TO BE TESTED
PARAMD   / /*ABS*/V.N.COSRCF/V.N.RCF $GETS ABSOLUTE VALU OF COS (ROW.COL) TERM
PARAMD   //*LE*/ /V.Y.KAPPA/V.N.COSRCF////V.N.LICHK $LICHK =-1 IF KAPPA < COSRC
PARTN    LIVECI. .SCADJ/CDUM. . ./+7/+1 $ HAVE BUBL VEC TO TRACK.TRIM TRAIL ZER
MERGE    CDUM. . . . .SCVEC/LIVECJ/+7/+1 $BUBBLE INCREMENTED AWAY FROM HEAD
COND     FALBOOK.LICHK $ CATALOG FAILURE POSITION
JUMP     MORCLI $ SKIP AROUND CATALOGING IF TEST WAS SUCCESSFUL
LABEL    FALBOOK $
PARAM    / /*MPY*/V.N.NORFAL/V.N.LICHK/ -1 $ SETS SIGNAL ONLY WHEN A COL FAILS.
$$                       HAS OPPOSITE SIGN TO LICHK.  POSSIBLE REPEATS ARE O.K.
PRTPARM  / /0/C.N.ROCNT $ ROW # OF CANDIDATE WHICH FAILED LI TEST
PRTPARM  / /0/C.N.CLCNT $ COL # OF CANDIDATE WHICH FAILED LI TEST
ADD      FALCCLI.LIVECJ/FALCCLJ/ $ ACCUM OF COL POS'NS OF FAILURES
```

BEST COPY
AVAILABLE

```
SWITCH   FALCCLI.FALCCLJ/ / V.N.LICHK $
LABEL    MORCLI $ CONTINUE LI TESTING IN THIS ROW EVEN AFTER A COL FAILS
SWITCH   LIVECI,LIVECJ/ / -1 $
PARAM    / /*EQ*/V.N.LICDUN/V.N.CLCNT/V.N.ZCOL $ LICDUN = -1 IF CLCNT = ZCOL
COND     NUROW.LICDUN $ JUMP OUTSIDE OF COL LOOP IF @ LAST COL
REPT     LICLUP,999 $$$$$$$$$$$ END OF COLUMN LOOP
$
LABEL    NUROW $
$
COND     GUDRO.NORFAL $ JUMP IF NO COLS IN CURRENT ROW HAD A LI FAILURE
ADD      FALRRLI, SCVECI/FALRRLJ/ $ ACCUMULATED ROW POSN'S OF FAILED ROWS
SWITCH   FALRRLI,FALRRLJ/ / -1 $ SWITCH ONLY IF THIS ROW FAILED.ELSE STAYS
LABEL    GUDRO $
PARAM    / /*SUB*/V.N.ROTEST/V.N.ZCOL/1 $ DECREMENT ZCOL BY ONE FOR ROTEST
PARAM    / /*EQ*/V.N.LIRDUN/V.N.ROCNT/V.N.ROTEST $ LIRDUN = -1.IF ROCNT=ROTEST
COND     KLENUP.LIRDUN $
JUMP     MOROW $ GO AROUND CLEAN UP IF MORE ROWS REMAIN TO BE TESTED
LABEL    KLENUP $
PARAML   FALCCLI/ /*TRAILER*/C.N.6/V.N.FALCDENS $ DENSITY OF THE COL FAIL VEC
PARAML   FALRRLI/ /*TRAILER*/C.N.6/V.N.FALRDENS $ DENSITY OF THE ROW FAIL VEC
PARAM    / /*LE*/V.N.DENSLK/V.N.FALCDENS/V.N.FALRDENS $ IF COL DENS </= ROW DEN.
$$                DENSITY SELECTION PARAMETER IS NEGATIVE
PRTPARM  / /C.N.0/C.N.DENSLK $DB
COND     OTHER.DENSLK $ SWITCH MODPARTN TO THE VECTOR W LWR DENSITY
SWITCH   FALRRLI.MODPARTN/ /-1 $
JUMP     MODSET $
LABEL    OTHER $
SWITCH   FALCCLI.MODPARTN/ /-1 $
LABEL    MODSET $
PARTN    FMOD.MODPARTN, /PHI, , , /+7/+2 $SURVIVORS OF LI TEST
PARTN    SCALF, .MODPARTN/PFVEC.../+7/+1 $ CLUSTER VECTOR FOR MERGING FUND-
$$                                      AMENTAL MODES WITH HARMONICS
```

```
MERGE     PFVEC,,,,,LONGNULL/HEADPF/+7/+1 $PUT PFVEC AT THE HEAD OF A LONG VEC
PARTN       LONGONE, ,HEADPF/SHORTONE,,,/+7/+1  $  PARTITION  LONGONE  DOWN   B
PERMANENT
MERGE     SHORTONE,,,,,LONGNULL/HEDSHORT/+7/+1  $APPEND  PERMANENT-SIZE  NULL  T
TAIL
ADD       HEDSHORT,LONGONE/NEGTAL/(1.0,0.0)/(-1.0,0.0) $PERMANENT-SIZE NEG @ TAI
$
$  PROVIDE FOR THE POSSIBILITY OF THE CULLING VECTOR CONTAINING 1'S IN THE END
$  POSITIONS, WHICH WOULD DESTROY THE FUNCTION OF THE SHIFTING VECTORS. CONVERT
$  SCALF TO ALL ONES.
$
MERGE     SCALF,,,,,SCADJ/COLSCAL/+7/+2/+2 $ SETS TRAILER TO RECTANGULAR
TRNSP     COLSCAL/SCALPRO $ CONVERT COL TO ROW
MPYAD     COLSCAL,SCALPRO, /SQUID/0 $SQUARE MTX OF READ D.P. IN PREP FOR DIAG
DIAGONAL SQUID/FULU/*COLUMN*/0.0 $FULU IS A CLUSTER OF ALL 1'S. LGTH=FMOD
PARTN     FULU,,MODPARTN/FUNPART,TOSS,,/+7/+1/+2/+2 $LGTH  1ST=PHI,TOSS=COMP  WR
FMOD
MERGE     FUNPART,,,,,FULU/HEDCLUS/+7/+1 $FORM CLUSTER OF FUNPART @ VEC HEAD
MERGE     TOSS,,,,,FULU/HEDTOSS/+7/+1/+2 $FORM CLUSTER OF TOSS @ VEC HEAD
ADD        FULU,HEDTOSS/TALCLUS/(1.0,0.0)/(-1.0,0.0)  $  CAP  ZEROES  @  HEAD  O
FUNPART
$
COND      ADJWRAP,ADJCYC $
$
COPY      PHI/PHIPI/ 0 $
ADD       NEGTAL, /LONGPRMI/(-1.0,0.0) $ TAIL CLUSTER = PERMANENT PRIMARY MODES
PARTN     SCVEC, ,HEDCLUS/,HMHED,,/+7/+1 $ HEAD SHIFTER TRIMMED TO LGTH = PHIPI
PARTN     SCADJ, ,TALCLUS/,HMTAL,,/+7/+1 $ TAIL SHIFTER TRIMMED TO LGTH = PHIPI
JUMP      ADJKUNT $
LABEL     ADJWRAP $
COPY      PHI/PHIAI/ 0 $
ADD       NEGTAL, /LONGARMI/(-1.0,0.0) $ TAIL CLUSTER = PERMANENT ADJOINT MODES
```

```
PARTN    SCVEC, ,HEDCLUS/,ADJHED,,/+7/+1 $ SHIFTER TRIMMED TO LGTH = PHIAI
PARTN    SCADJ, ,HEDCLUS/,ADJTAL,,/+7/+1 $ SHIFTER TRIMMED TO LGTH = PHIAI
$
JUMP     HARMONY $
$
LABEL    MOROW $
PARAM    / /*MPY*/V,N,CLCNT/1/0 $ RESET COLUMN COUNT TO ZERO
PARTN    SCVECI, ,SCADJ/RDUM, , , /+7/+1 $ TRIM TRAILING ZERO
MERGE    RDUM, , , , ,SCVEC/SCVECJ/+7/+1 $ BUBBLE INCREMENTED AWAY FROM HEAD
SWITCH   SCVECI,SCVECJ/ / -1 $
COPY     SCVECI/LIVECI/ 0 $ COL BUBBLE INDEX ALIGNED WITH ROW TRACKER
$
REPT      LIRLUP,999 $
$
LABEL    ADJKUNT $
PARAM    / /*MPY*/V,N,ADJCYC/1/-1 $
$
REPT      ADJLUP,999 $
$
LABEL    HARMONY   $
$
OUTPUT1, ,,,,/ /-1/0 $ CALLS THE DEFAULT LABEL. NEEDED FOR REWINDS LATER.
OUTPUT1  PHIP1,PHIAI,SOLP,SOLA, / /0/0 $MANY CALLS TO BE MADE IN HARMONIC PHASE
OUTPUT1, ,,,,/ /-1/1 $ SETS THE DEFAULT LABEL
OUTPUT1  HMHED,HMTAL,ADJHED,ADJTAL, / /0/1 $
OUTPUT1, ,,,,/ /-1/2 $ SETS THE DEFAULT LABEL
OUTPUT1  LONGNULL,LONGPRMI,LONGARMI,LONGONE, / /0/2 $
$
ALTER 154,154 $ REMOVE OPTIMIZATION LOOP TO PREVENT O'FLOW OF CEITBL
$
ENDALTER $ END OF ALTER FOR RITZ FUNDAMENTAL MODES
```

APPENDIX B

DMAP CODING FOR PRIMARY AND ADJOINT HARMONICS

```
APP  DMAP $ FOR EXECUTION AFTER RITZFUND. INPUTS FROM INPT,INP1,INP2
BEGIN $  PROGRAMMED FOR 1988 VERSION OF NASTRAN. OUTPUT TO INP3
FILE LONGPRMI=SAVE/LONGARMI=SAVE/PGENI=SAVE/AGENI=SAVE/FULU=SAVE $
FILE PHHED=SAVE/PHTAL=SAVE/AHHED=SAVE/AHHED=SAVE/AHTAL=SAVE $
PARAM    / /*MPY*/V,N,PRIMCYC/+1/-1 $ CONTROL PARAM FOR PRIMARY 1ST HARMONIC
PARAM    / /*ADD*/V,N,ADJCYC/+1/ 0 $ CONTROL PARAM FOR ADJUNCT 1ST HARMONIC
PARAM    / /*ADD*/V,N,NUPGEN/+1/ 0 $ CONTROL PARAM FOR PRIMARY HIGHER HARMONICS
PARAM    / /*ADD*/V,N,NUAGEN/+1/ 0 $ CONTROL PARAM FOR ADJUNCT HIGHER HARMONICS
PARAM    / /*MPY*/V,N,HARMNO/ 1/ 0 $ SET THE HARMONIC COUNTER TO ZERO.
$
LABEL    HMNICGEN $ TOP OF LOOP FOR HARMONIC GENERATION %%%%%%%%%%%%%
$
PARAM    / /*ADD*/V,N,HARMNO/V,N,HARMNO/ 1 $INCREMENT THE HARMONIC COUNT BY ONE
COND     PRIMPREP,PRIMCYC $
COND     ADJPREP,ADJCYC $
COND     PHMNPREP,NUPGEN $
COND     AHMNPREP,NUAGEN $
LABEL    PRIMPREP $
INPUTT1  /, , , , /-1/0 $
INPUTT1  /HLV,... /0/0 $ READ PHIPI INTO HLV
COPY     HLV/PHIPI/ 0 $ THIS IS THE ROOT FOR THE 1ST MERGE OF HARMONICS TO PRIM
EQUIV    HLV,TESTER/PRIMCYC  $ HLV AND TESTER ARE INTERNAL NAMES OF GENERATOR.
$$       LATER ON, EQUIV WILL BE BROKEN AT TOP'O LOOP.
INPUTT1  /KMMTX,,,, /1/0 $ SKIP PASSED 2ND DB AND READ SOLP INTO KMMTX
INPUTT1  /, , , , /-1/1 $
INPUTT1  /HEDVECI,TALVECI,,, /0/1 $READ HMHED INTO HEDVECI & HMTAL INTO TALVECI
```

RITZ METHOD FOR TRANSIENT RESPONSE IN SYSTEMS HAVING UNSYMMETRIC STIFFNESS

```
INPUTT1  /, , , , /-1/1 $
INPUTT1  /PGENI,PHTAL,,,/0/1 $READ HMHED INTO PGENI;READ HMTAL INTO PHTAL.
COPY     PGENI/PHHED/ 0 $ DUMMY STATEMENT TO FOOL THE COMPILER
ADD      HEDVECI,TALVECI/FULU $ FLUFF TO HELP FIAT LOCATE THE REAL FULU
JUMP     HMYBUS $ GO AROUND THE AJOINT PREP
LABEL    ADJPREP $
INPUTT1  /,,,,/-1/0 $ REWIND FROM PREVIOUS PASS THRU LOOP AND POSITION @ 1ST DB
INPUTT1  /HLV,,,, /1/0 $SKIP PASSED 1ST DB AND READ PHIAI INTO HLV
COPY     HLV/PHIAI/0 $THIS IS THE ROOT FOR THE 1ST MERGE OF HARMONICS TO ADJOIN
EQUIV    HLV,TESTER/ADJCYC  $ HLV & TESTER ARE INTERNAL NAMES OF GENERATOR
INPUTT1  /KMMTX,,,, /1/0 $ SKIP PASSED 3RD DB AND READ SOLA INTO KMMTX
INPUTT1  /, , , , /-1/1 $
INPUTT1  /HEDVECI,TALVECI,,,/2/1 $READ ADJHED INTO HEDVECI & ADJTAL INTO TALVEC
INPUTT1  /, , , , /-1/1 $
INPUTT1  /AGENI,AHTAL,,,/2/1 $SKIP 2DB & READ ADJHED > AGENI;READ ADJTAL >PHTAL.
COPY     AGENI/AHHED/ V,N,ADJCYC $
JUMP     HMYBUS $
LABEL    PHMNPREP $
COPY     PGENI/HLV/ 0 $
EQUIV    PGENI,TESTER/NUPGEN  $ TESTER IS INTERNAL NAME OF GENERATOR
INPUTT1  /,,,, /-1/0 $ REWIND FROM PREVIOUS PASS THRU LOOP AND POSITION @ 1ST D
INPUTT1  /KMMTX,,,, /2/0 $ SKIP PASSED 1ST 2 DB'S AND READ SOLP INTO KMMTX
EQUIV    PHHED,HEDVECI/NUPGEN/PHTAL,TALVECI/NUPGEN $
JUMP     HMYBUS $
LABEL    AHMNPREP $
COPY     AGENI/HLV/ 0 $
EQUIV    AGENI,TESTER/NUAGEN  $ TESTER IS INTERNAL NAME OF GENERATOR
INPUTT1  /,,,, /-1/0 $ REWIND FROM PREVIOUS PASS THRU LOOP AND POSITION @ 1ST D
INPUTT1  /KMMTX,,,, /3/0 $ SKIP PASSED 1ST 3 DB'S AND READ SOLA INTO KMMTX
EQUIV    AHHED,HEDVECI/NUAGEN/AHTAL,TALVECI/NUAGEN $
LABEL    HMYBUS $
```

RITZ MODES FOR UNSYMMETRIC MATRICES
DMAP CODING OF THE THEORY

```
PARAM    / /*MPY*/V,N,INITIAL/+1/-1 $ NEWLY GENERATED CANDIDATE GOING TO
$$                                     INITIAL TEST.  SET TO -1.
MPYAD    KMMTX,HLV, /UHC/0/1/0/2 $ CANDIDATES OF THE INERTIAL RESPONSES
PURGE    KMMTX $
COPY     UHC/CLONUHC/ 0 $
MPYAD    CLONUHC,UHC, /UHCSQ/ +1/+1/0/2 $
DIAGONAL UHCSQ/SCALH/*DIAGONAL*/-0.5 $ VECTOR OF EUCLIDEAN LENGTHS.
MPYAD    UHC,SCALH, /PHIHC/0/1/0/2 $ MATRIX OF CANDIDATE FIRST HARMONIC MODES.
COPY     PHIHC/CANDIDAT/ 0 $ USE GENERALIZED LOOP NAMES
$$
LABEL    LIPREP $ TOP OF LOOP FOR LINFAR INDEPENDENCE CHECKING %%%%%%%%%%%%%
PARAM    / /*MPY*/V,N,RFALNO/1/-1 $ SET DEFAULT TO NEGATIVE.
$$
MPYAD    CANDIDAT,TESTER, /CVST/ +1/+1/0/2 $ HARMONICS AGAINST THE GENERATORS.
MATPRN   CVST,,,,/ / $
COND     RECT0,INITIAL $ SUBSTITUTE A REDUCED FULU IF REMNANT < GENERATOR
DIAGONAL CVST/FULU/*COLUMN*/0.0 $ALL ONE VEC FOR HARM VS HARM (CVST IS SQUARE)
LABEL    RECT0 $
EQUIV    HLV,TESTER/MODSPEC $BREAK EQUIV W TESTER
EQUIV    PGENI,TESTER/MODSPEC/AGENI,TESTER/MODSPEC $BREAK EQUIV W TESTER
COPY     HEDVECI/HMROWI/ 0 $ ROW BUBBLE STARTING FROM HEAD
ADD      HMROWI,HEDVECI/FALHRI/ /(-1.0,0.0) $ NULL. SAME LENGTH AS HEDVECI
COPY     FALHRI/FALHCI/ 0 $ NULL. SAME LENGTH AS HEDVECI
PARAM    / /*MPY*/V,N,CLKNT/ 1/ 0 $ RESET COL COUNT TO 0 BEFORE LI CHECK
PARAM    / /*MPY*/V,N,ROKNT/ 1/ 0 $ RESET ROW COUNT TO 0 BEFORE LI CHECK
PARAML   CVST//*TRAILER*/2/V,N,HROW $ ROWS IN CVST
COND     RECTT,INITIAL $ JUMP IF ON RECTANGULAR CYCLE
PARAM    / /*SUB*/V,N,HROW/V,N,HROW/1 $ REDUCE ROW TEST VALUE 1 FOR TRIANGLE
LABEL    RECTT $
PARAML   CVST//*TRAILER*/1/V,N,HCOL $ COLS IN CVST
$
LABEL    HLIRLUP $ TOP OF ROW PORTION OF LINEAR INDEPENDENCE CHECK %%%%%%%%%%%
```

RITZ MODES FOR UNSYMMETRIC MATRICES
DMAP CODING OF THE THEORY

```
$
COPY     HEDVECI/HMCOLI/ 0 $ COL BUBBLE STARTING FROM HEAD
PARTN    CVST, ,HMROWI/ ,ROCH, , /+7/+2 $
PARAM    / /*ADD*/V,N,ROKNT/V,N,ROKNT/ 1 $ ROW COUNT MONITOR INCREMENTED BY ONE
COND     RECT1,INITIAL $
PARAM    / /*MPY*/V,N,CLKNT/1/V,N,ROKNT $ SET COL COUNT=ROW COUNT IF TRIANGLE
LABEL    RECT1 $
$
LABEL    HLICLUP $ TOP OF COLUMN PORTION OF LI LOOP %%%%%%%%%%%%%%%%%%%%%%%%%%
$
PARAM    / /*ADD*/V,N,CLKNT/V,N,CLKNT/ 1 $ INCREMENT THE COLUMN COUNT
SCALAR   ROCH/ /1/V,N,CLKNT/ /V,N,RCH $ RCH IS DBL PREC FETCH OF COSINE
$$ TERM IN ROW POSITION INDEXED BY THE CONSTANT PARAMETER '1' AND IN THE COL
$$ POSITION INDEXED BY THE VARIABLE PARAMETER CLKNT.
PARAMD   / /*ABS*/V,N,COSRCH/V,N,RCH $ GETS ABS. VAL. OF COS(ROW,COL) TERM
PARAMD   / /*LE*/ /V,Y,KAPPA/V,N,COSRCH/ / / /V,N,LIHZK $
COND     CATALOG,LIHZK $ GO TO CATALOGING IF LIHZK IS NEGATIVE
JUMP     MORHCOL $ JUMP TO MORE COL PROCESSING IF TEST PASSED
LABEL    CATALOG $
COND     RECT2,INITIAL $
PARAM    / /*MPY*/V,N,RFALNO/V,N,LIHZK/ -1 $ SETS SIGNAL ONLY WHEN A COL FAILS.
$$                    RFALNO TAKES ON OPPOSITE SIGN TO LIHZK. REPEATS ARE OK.
LABEL    RECT2 $
PRTPARM  / /0/C,N,ROKNT $ ROW # OF CANDIDATE TERM WHICH FAILED LI TEST
PRTPARM  / /0/C,N,CLKNT $ COL # OF CANDIDATE TERM WHICH FAILED LI TEST
COND     RECT3,INITIAL $
ADD      FALHCI,HMCOLI/FALHCJ/ $ ACCUMULATION OF COL POS'NS OF FAILURES
SWITCH   FALHCI,FALHCJ/ /V,N,LIHZK  $
$
```

RITZ MODES FOR UNSYMMETRIC MATRICES
DMAP CODING OF THE THEORY

```
LABEL    MORHCOL $ CONTINUE LI TESTING IN THIS ROW EVEN AFTER A COL FAILS
$
PARAM    / /*EQ*/V,N,LICDON/V,N,CLKNT/V,N,HCOL $ LICDON = -1 IF CLKNT = HCOL
COND     GNROW,LICDON $ JUMP OUTSIDE OF COL LOOP IF @ LAST COL.
PARTN    HMCOLI, ,TALVECI/DUMMY,,,/+7/+1 $ TRIM TRAILING ZERO
MERGE    DUMMY, , , , ,HEDVECI/HMCOLJ/+7/+1 $ BUBBLE INCREMENTED AWAY FROM HEAD
SWITCH   HMCOLI,HMCOLJ/ / -1 $
REPT     HLICLUP,999 $ $$$$$$$$$$$$$$$  END OF COLUMN LOOP
$
LABEL    GNROW $ CONSIDER TESTING ANOTHER ROW
$
COND     GODRO,RFALNO $ JUMP IF NO COLS IN CURRENT ROW HAD A LI FAILURE
LABEL    RECT3 $ CATALOG ROW FAILURE
ADD      FALHRI, HMROWI/FALHRJ/ $ ACCUMULATED ROW POS'NS OF FAILED ROWS
SWITCH   FALHRI,FALHRJ/ / -1 $ SWITCH ONLY IF THIS ROW FAILED, ELSE STAYS
COND     RECT4,INITIAL $ BYPASS IF ON RECTANGULAR ROUTE
PARAM    / /*MPY*/V,N,RFALNO/V,N,RFALNO/-1 $ RESET TO NEGATIVE.
LABEL    RECT4 $
$
LABEL    GODRO $
$
PARAM    / /*EQ*/V,N,LIRDON/V,N,ROKNT/V,N,HROW $ LIRDON = -1 IF ROKNT = HROW
COND     CLENUP,LIRDON $ JUMP OUT OF LI CHECKING IF MTX IS COMPLETELY EXAMINED
PARAM    / /*MPY*/V,N,CLKNT/1/0 $ RESET COLUMN COUNT TO ZERO
PARTN    HMROWI, ,TALVECI/DUMMY,,,/+7/+1 $ TRIM TRAILING ZERO
MERGE    DUMMY,,,,,HEDVECI/HMROWJ/+7/+1 $BUBBLE INCREMENTED AWAY FROM HEAD
SWITCH   HMROWI,HMROWJ/ /-1 $
$
REPT      HLIRLUP,999 $ END OF ROW PORTION OF LINEAR INDEPENDENCE LOOP%%%%%%%%%
$
```

RITZ MODES FOR UNSYMMETRIC MATRICES
DMAP CODING OF THE THEORY

```
LABEL    CLENUP $
$
COND     RECT5,INITIAL $
PARAML   FALHCI/ /*TRAILER*/C,N,6/V,N,DENSFALC $ DENSITY OF THE COL FAIL VEC
PARAML   FALHRI/ /*TRAILER*/C,N,6/V,N,DENSFALR $ DENSITY OF THE ROW FAIL VEC
PARAM    / /*LE*/V,N,SLKDENS /V,N,DENSFALC/V,N,DENSFALR $ IF COL DENS
$$       </= ROW DENS, THE DENSITY SELECTION PARAMETER IS NEGATIVE.
COND     UTHER,SLKDENS $SET MODEPART TO THE VECTOR WITH LWR DENSITY
LABEL    RECT5 $
COPY     FALHRI/MODEPART/ 0 $
JUMP     MODESET $
LABEL    UTHER $
COPY     FALHCI/MODEPART/ 0 $
LABEL    MODESET $
PARAML   MODEPART/ /*TRAILER*/C,N,6/V,N,MODENSY $ DENSITY OF MODEPART
PRTPARM  / /0/C,N,MODENSY $
PARAM    / /*GE*/V,N,FILLED/V,N,MODENSY/10000 $ FILLED=-1 IF MODEPART IS FULL
COND     FOLD,FILLED $
JUMP     FLEDGE  $
LABEL    FOLD $ SAVE AND GO
PARAM      /  /*GT*/V,N,SUMHUM/V,N,HARMNO/2  $  SUMHUM=-1  IF 1ST HARMS OF P & .
PASSED
PRTPARM  / /0/C,N,SUMHUM $
COND     ORTHOG,SUMHUM $ IF 1ST HARMS FAIL RESTORE ORIGINAL NAMES TO OUTPUT
INPUTT1  /, , , , /-1/0 $
INPUTT1  /PHIPI,PHIAI,,,/0/0 $ FUNDS W/O HARMONICS
INPUTT1  /, , , , /-1/1 $
INPUTT1  /HEADPHI,TAILPHI,,,/0/1 $TRACKERS W/O HMNIC.HMHED=HEADPHI.HMTAL=TAILPH.
JUMP     NOBIZNEZ $ COPY OUT AS THEY CAME IN
```

BEST COPY
AVAILABLE

RITZ MODES FOR UNSYMMETRIC MATRICES
DMAP CODING OF THE THEORY

```
LABEL    FLEDGE $
PARTN    CANDIDAT,MODEPART,/REMNANT,,,/+7/+2 $SURVIVORS OF LI TEST
$$ PROVIDE FOR THE POSSIBILITY OF THE CULLING VECTOR CONTAINING 1'S IN THE
$$ ENDS, WHICH WOULD DESTROY THE SHIFTING VECTORS. CONVERT SCALH TO ALL ONES.
COND     RECT6,INITIAL $
JUMP     TRIA6 $
LABEL    RECT6 $
MERGE    SCALH,,,,,TALVECI/COLSCAL/+7/+2/+2 $ SETS TRAILER TO RECTANGULAR
TRNSP    COLSCAL/SCALPRO $ CONVERT COL TO ROW
MPYAD    COLSCAL,SCALPRO, /SQUID/0 $ SQAURE MTX OF REAL S.P. IN PREP FOR DIAG
DIAGONAL SQUID/FULU/*COLUMN*/0.0 $FULU IS A CLUSTER OF ALL ONES LGTH=CANDIDAT
LABEL    TRIA6 $
PARTN    FULU,,MODEPART/HMYPART,TOSS,,/7/1/2/2 $1'S.LGTH 1ST =REMNANT.TOSS=COMP
MERGE    HMYPART,,,,,FULU/HEDCLUS/+7/+1/2 $FORM CLUSTER OF HMYPART @ VEC HEAD
PARTN    HEDVECI,,HEDCLUS/,HEDVECJ,,/7/1/2/2 $NU SHIFTER HAS LGTH=REMNANT
MERGE    TOSS,,,,,FULU/HEDTOSS/7/1/2 $FORM CLUSTER OF TOSS @ VEC HEAD
ADD      FULU,HEDTOSS/TALCLUS/(1.0,0.0)/(-1.0,0.0) $CAP ZEROES @ HEAD OF HMYPAR
PARTN    TALVECI,,TALCLUS/,TALVECJ,,/7/1/2/2 $NU SHIFTER HAS LGTH=REMNANT
SWITCH   HEDVECI,HEDVECJ/ / -1 $
SWITCH   TALVECI,TALVECJ/ / -1 $
PRTPARM  / /0/C,N,INITIAL $
COND     TEST2,INITIAL $ If INITIAL IS NEGATIVE GO TO 2nd LI TEST
JUMP     DEPOT $READY FOR MERGING AND GENERATING (1.ADJ 2.NU PHMNY 3.NU AHMNY)
LABEL    TEST2 $
$Test whether one or less columns of REMNANT are left. SET PARAMETER IF SO.
PARAML   REMNANT/ /*TRAILER*/1/V,N,REMCOL $
PARAM    / /*LE*/V,N,SALVAGE/V,N,REMCOL/1 $
COND     DEPOT,SALVAGE $
SWITCH   REMNANT,CANDIDAT/ /V,N,INITIAL $
COPY     CANDIDAT/TESTER/V,N,INITIAL $
PARAM    / /*MPY*/V,N,INITIAL/V,N,INITIAL/ -1 $
REPT      LIPREP,1 $ MAKE A 2ND PASS FOR THE LI TESTS ON HARMONICS ALONE.
```

```
$
LABEL    DEPOT $ STAGING POINT.WRAPUP.MERGE.HARMONIC GENERATION.
COND     QUIV1,PRIMCYC $
JUMP     CHKADJ $
LABEL    QUIV1 $
INPUTT1  /, , , , /-1/2 $
INPUTT1  /LONGPRMI,,,, /1/2 $SKIP 1 DB & READ IN LONGPRMI
EQUIV    LONGPRMI,LONGRMI/PRIMCYC/PHIPI,PHII/PRIMCYC $
JUMP     MERGBUS $
LABEL    CHKADJ $
COND     QUIV2,ADJCYC $
JUMP     CHKPGEN $
LABEL    QUIV2 $
INPUTT1  /, , , , /-1/2 $
INPUTT1  /LONGARMI,,,, /2/2 $SKIP 2 DB & READ IN LONGARMI
EQUIV    LONGARMI,LONGRMI/ADJCYC/PHIAI,PHII/ADJCYC $
JUMP     MERGBUS $
LABEL    CHKPGEN $
COND     QUIV3,NUPGEN $
JUMP     CHKAGEN $
LABEL    QUIV3 $
EQUIV    LONGPRMI,LONGRMI/NUPGEN/PHIPI,PHII/NUPGEN $
JUMP     MERGBUS $
LABEL    CHKAGEN $
COND     QUIV4,NUAGEN $
JUMP     MERGBUS $
LABEL    QUIV4 $
EQUIV    LONGARMI,LONGRMI/NUAGEN/PHIAI,PHII/NUAGEN $
LABEL    MERGBUS $
INPUTT1  /, , , , /-1/2 $
INPUTT1  /LONGNULL,,,,/0/2 $READ IN LONGNULL
INPUTT1  /LONGONE,,,,/2/2 $ SKIP 2 DB & READ IN LONGONE
```

BEST COPY
AVAILABLE

RITZ MODES FOR UNSYMMETRIC MATRICES
DMAP CODING OF THE THEORY

```
MERGE    HMYPART,,,,,LONGNULL/LHRMHED/+7/+1 $ INCREMENT = LENGTH OF NEW HARMONIC
PARTN    LONGONE, ,LHRMHED/DUMMY,,,/+7/+1 $ PARTITION LONGONE DOWN BY HARMONIC
MERGE    DUMMY,,,,,LONGNULL/HEDSHRTH/+7/+1 $APPEND HARMONIC-SIZE NULL TO TAIL
ADD      LONGONE,HEDSHRTH/LHRMTAL/ /(-1.0,0.0) $HARMONIC-SIZE CLUSTER @ TAIL
PARTN    LONGRMI,,LHRMHED/DUMMY,,,/+7/+1 $TRIM A HARMONIC INCREMENT OF ZEROES
$$         FROM THE HEAD OF LONGRMI.
MERGE    DUMMY,,,,,LHRMTAL/BBLRM/+7/+1 $PLUG A HARMONIC INCREMENT OF ZEROES
$$         ONTO THE TAIL OF LONGRMI.
ADD      BBLRM,LHRMTAL/LONGRMJ/ $ CLUSTER=NUHARM INC + ACCEPTED
PARTN    LHRMTAL,,LONGRMJ/ ,TRIMRM,,/+7/+1 $ THIS IS THE PARTITIONING VECTOR
$$         NEEDED FOR COMBINING THE HARMONIC TO THE MATRIX OF ACCEPTED MODES
$Test whether one or less columns of REMNANT are left. SET PARAMETER IF SO.
$
PARAML   REMNANT/ /*TRAILER*/1/V,N,REMCOL $
MERGE    PHII, ,REMNANT,,TRIMRM,/PHIJ/+7/+2 $ MERGED!!
SWITCH   LONGRMJ,LONGPRMI/ /V,N,PRIMCYC $
SWITCH   LONGRMJ,LONGPRMI/ /V,N,NUPGEN $
SWITCH   LONGRMJ,LONGARMI/ /V,N,ADJCYC $
SWITCH   LONGRMJ,LONGARMI/ /V,N,NUAGEN $
SWITCH   PHIJ,PHIPI/ /V,N,PRIMCYC $
SWITCH   PHIJ,PHIPI/ /V,N,NUPGEN $
SWITCH   PHIJ,PHIAI/ /V,N,ADJCYC $
SWITCH   PHIJ,PHIAI/ /V,N,NUAGEN $
SWITCH   HEDVECI,PHHED/ /V,N,PRIMCYC $
SWITCH   HEDVECI,AHHED/ /V,N,ADJCYC $
SWITCH   HEDVECI,PHHED/ /V,N,NUPGEN $
SWITCH   HEDVECI,AHHED/ /V,N,NUAGEN $
SWITCH   TALVECI,PHTAL/ /V,N,PRIMCYC $
SWITCH   TALVECI,AHTAL/ /V,N,ADJCYC $
SWITCH   TALVECI,PHTAL/ /V,N,NUPGEN $
SWITCH   TALVECI,AHTAL/ /V,N,NUAGEN $
$
```

RITZ MODES FOR UNSYMMETRIC MATRICES
DMAP CODING OF THE THEORY

```
PARAM    //*EQ*/V,N,HARMDONE/C,Y,MODSPEC/V,N,HARMNO $ IF # HARM=MODSPEC =>DONE
COND     ORTHOG,HARMDONE $JUMP OUTSIDE HARMONIC LOOP IF HARMONICS ARE DONE
$
COND     PRIMOUT,PRIMCYC $
JUMP     ADJHRMNY $
LABEL    PRIMOUT $
SWITCH   REMNANT,PGENI/ /V,N,PRIMCYC $
PARAM    //*MPY*/V,N,PRIMCYC/V,N,PRIMCYC/ -1 $ RESET PRIMCYC TO POSITIVE
PARAM    //*MPY*/V,N,ADJCYC/+1/-1 $ ENABLE THE LOOP FOR THE ADJOINT 1ST HARM
JUMP     HLOOPEND $
LABEL    ADJHRMNY $
COND     ADJOUT,ADJCYC $
JUMP     PHMNZ $
LABEL    ADJOUT $
SWITCH   REMNANT,AGENI/ /V,N,ADJCYC $-------
PARAM    //*MPY*/V,N,ADJCYC/V,N,ADJCYC/ -1 $ RESET ADJCYC TO POSITIVE
PARAM    //*MPY*/V,N,NUPGEN/+1/-1 $ ENABLE THE LOOP FOR THE PRIM HIGHER HARM
JUMP     HLOOPEND $
LABEL    PHMNZ $
COND     PHMNOUT,NUPGEN $
JUMP     AHMNZ $
LABEL    PHMNOUT $
SWITCH   REMNANT,PGENI/ /V,N,NUPGEN $--------
PARAM    //*MPY*/V,N,NUPGEN/V,N,NUPGEN/ -1 $ RESET NUPGEN TO POSITIVE
PARAM    //*MPY*/V,N,NUAGEN/+1/-1 $ ENABLE THE LOOP FOR THE ADJ HIGHER HARM
JUMP     HLOOPEND $
LABEL    AHMNZ $
SWITCH   REMNANT,AGENI/ /V,N,NUAGEN $-------
PARAM    //*MPY*/V,N,NUAGEN/V,N,NUAGEN/ -1 $ RESET NUAGEN TO POSITIVE
PARAM    //*MPY*/V,N,NUPGEN/+1/-1 $ ENABLE THE LOOP FOR THE PRIM HIGHER HARM
LABEL    HLOOPEND $
$
```

RITZ MODES FOR UNSYMMETRIC MATRICES
DMAP CODING OF THE THEORY

```
PARAM    / /*LE*/V,N,TESTOVER/V,N,REMCOL/1 $ TESTOVER =-1 IF  REMCOL ((/=) 1
$$                               THIS IS THE EXIT  IN CASE FEWER HARMONICS PASS
$$                               THE LI TEST THAN ARE SPECIFIED BY MODSPEC
COND     ALTCHK,TESTOVER $ EXIT WHEN HARMONICS ARE EXHAUSTED PRIOR TO MODSPEC
JUMP     USUAL  $
LABEL    ALTCHK $
COND     USUAL,ADJCYC $ ADJOINT GETS A CHANCE TO GENERATE A SINGLE
COND     ORTHOG,NUPGEN $ PREVENT ANOTHER HARMONIC TO BE GENERATED FROM A SINGLE
COND     USUAL,NUAGEN $ ADJHRM GETS A CHANCE TO GENERATE A SINGLE
JUMP     ORTHOG $
$
LABEL     USUAL $
PURGE    KMMTX/MODSPEC $
REPT      HMNICGEN,999 $ END OF HARMONIC GENERATOR LOOP %%%%%%%%%%%%%%%%%%%%%%%
$
LABEL    ORTHOG $
$
MERGE    PHHED, , , , ,LONGNULL/LONGHED1/+7/+1 $**START OF HEADPHI CONSTRUCTION
PARTN    LONGONE, ,LONGHED1/DUMMY, , , /+7/+1 $ LUMP TO MERGE ON HEAD
MERGE    DUMMY, , , , ,LONGNULL/MISSTAIL/+7/+1 $ ONE MISSING FROM TAIL
ADD      LONGONE,MISSTAIL/LONGTAL1/ /(-1.0,0.0) $**START OF TAILPHI CONSTRUCTIO
PARTN    LONGONE, ,LONGPRMI/ ,DUMMY, ,/+7/+1 $ALL ONES OF LGTH=ACCEPTED VECTORS
MERGE    DUMMY, , , , ,LONGNULL/HEADER/+7/+1 $HEAD CLUSTER OBVERSE OF LONGPRMI
PARTN    LONGHED1, ,HEADER/ ,HEADPHI, , /+7/+1 $ SAVE FOR DELIVERY TO ORTHOG
PARTN    LONGTAL1, ,LONGPRMI/ ,TAILPHI, , /+7/+1 $ SAVE FOR DELIVERY TO ORTHOG
LABEL    NOBIZNEZ $ GET OUT WITH OUTPUT SAME AS INPUT
OUTPUT1, ,,,,/ /-1/3 $ SET DEFAULT LABEL
OUTPUT1  PHIPI,HEADPHI,TAILPHI,PHIAI, / /0/3 $
LABEL    PRINTOUT $
PRTPARM  //0/C,N,HARMNO $
LABEL    FINIS $
END   $ FINISH OF DMAP PROGRAM FOR RITZ HARMONICS
```

APPENDIX C

RITZRTHG.DMP
SELF AND DUO ORTHOGONALIZATION

```
NASTRAN MAXFILES = 60,FILES = (INP3,INP4)
APP       DMAP $ PROGRAMMED FOR 1988 NASTRAN. OUTPUT TO PUNCH FILE.
$$  EXECUTES AFTER BOTH RITZFUND AND RITZHARM TO ORTHOGONALIZE RITZ MODES
BEGIN $ $$ORTHOG.DMP
INPUTT1   /, , , , /-1/3 $
INPUTT1   /PHIPI,HEADPHI,TAILPHI,PHIAI, /0/3 $
PARTN     PHIPI,HEADPHI, / , ,PHI1, /+7/+2 $
COPY      PHI1/ZETA1/ 0 $
PARTN     HEADPHI, ,TAILPHI/DUMMY, , , /+7/+1 $
MERGE     DUMMY, , , , ,HEADPHI/BBLHI/+7/+1 $
PARTN     PHIPI,BBLHI, / , ,PHI2, /+7/+2 $
MPYAD     PHI1,PHI2, /NUM/+1/-1 $.
COPY      PHI1/CLONPHI1/ 0 $
MPYAD     CLONPHI1,PHI1, /DEN/+1/+1 $
SCALAR    NUM/ /1/1/ / /V,N,SPXNUM $
SCALAR    DEN/ /1/1/ / /V,N,SPXDEN $
PARAMR    / /*DIVC*/ / / /V,N,A11/V,N,SPXNUM/V,N,SPXDEN $
ADD       PHI1,PHI2/ZETA2/V,N,A11 $ SINGLE PREC.WON'T TAKE DBL PREC!!!
PARTN     TAILPHI, ,HEADPHI/DUMMD, , ,/+7/+1 $
MERGE     DUMMD,,,,,TAILPHI/BBLTI/+7/+1 $
ADD       BBLTI,TAILPHI/PTALCLUI/ $
PARTN     TAILPHI, ,PTALCLUI/,BUILDI, , /+7/+1 $
MERGE     ZETA1, ,ZETA2, ,BUILDI, /ZETAI/+7/+2 $
MPYAD     ZETAI,PHIPI, /COEFI/+1/+1 $
ADD       BBLHI,HEADPHI/PHEDCLUI/ $
PARAM     / /*ADD*/V,N,ROWCOW/2/0 $
PARAML    PHIPI/ /*TRAILER*/1/V,N,PCOL $
```

RITZ MODES FOR UNSYMMETRIC MATRICES
DMAP CODING OF THE THEORY

```
LABEL      ORTHLUP $ TOP OF SELF ORTHOGONALIZATION LOOP
PARAM      / /*ADD*/V,N,ROWCOW/V,N,ROWCOW/1 $
PARTN      COEFI,PHEDCLUI, / , ,CAI, /+7/+2 $.
PARTN      BBLHI, ,TAILPHI/DUMVEC, , , /+7/+1 $
MERGE      DUMVEC, , , , ,HEADPHI/BBLHJ/ +7/+1 $
PARTN      COEFI,BBLHJ, / , ,CFI, /+7/+2 $
SOLVE      CAI,CFI/AIN/-1/-1/2 $
PARTN      BBLTI, ,HEADPHI/DMY, , , /+7/+1 $
MERGE      DMY, , , , ,TAILPHI/BBLTJ/ +7/+1 $
ADD        PTALCLUI,BBLTJ/PTALCLUJ/ $
PARTN      TAILPHI, ,PTALCLUJ/ ,BUILDJ,,/+7/+1 $
PARTN      PTALCLUJ, ,TAILPHI/ ,UNIT, , /+7/+1/ /2 $ UNIT IS RECTANGULAR S.P.
MERGE      AIN,UNIT, , , ,BUILDJ/AJN/+7/+2 $
ADD        PHEDCLUI,BBLHJ/PHEDCLUJ/ $
PARTN      PHIPI,PHEDCLUJ, / , ,PHIZ, /+7/+2 $
MPYAD      PHIZ,AJN, /ZETAX/0$
MERGE      ZETAI, ,ZETAX, ,BUILDJ, /ZETAJ/+7/+2 $
MPYAD      ZETAJ,PHIPI, /COEFJ/+1/+1 $
SWITCH     ZETAI,ZETAJ/ / -1 $
SWITCH     BBLHI,BBLHJ/ / -1 $
SWITCH     PHEDCLUI,PHEDCLUJ/ / -1 $
SWITCH     BBLTI,BBLTJ/ / -1 $
SWITCH     PTALCLUI,PTALCLUJ/ / -1 $
SWITCH     BUILDI,BUILDJ/ / -1 $
SWITCH     COEFI,COEFJ/ / -1 $
PARAM      / /*EQ*/V,N,SELFDUN/V,N,ROWCOW/V,N,PCOL $
COND       DUALORTH,SELFDUN $
REPT       ORTHLUP,999 $
LABEL      DUALORTH $
COPY       ZETAI/CLONZETA/ 0 $
MPYAD      CLONZETA,ZETAI, /ZSQ/+1/+1 $
MATPRN     ZSQ,,,,// $
```

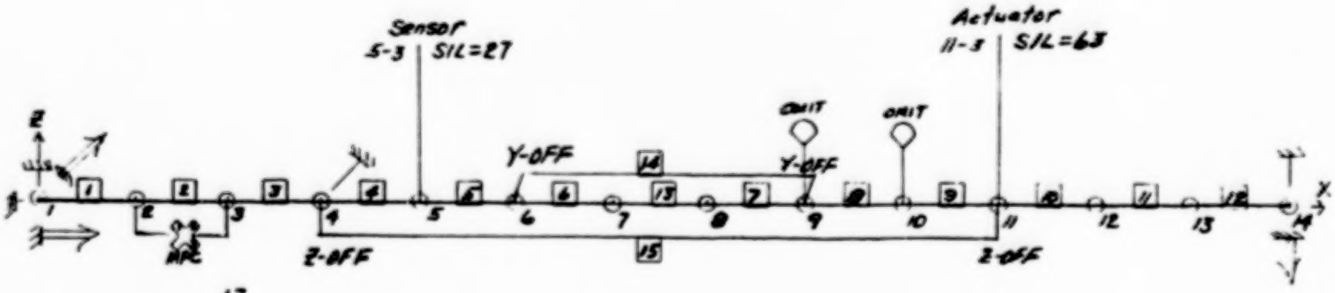
RITZ MODES FOR UNSYMMETRIC MATRICES
DMAP CODING OF THE THEORY

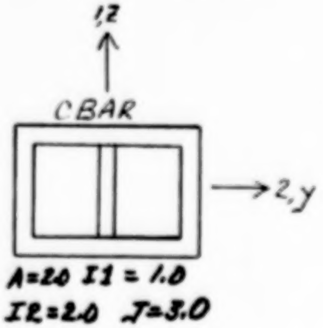
```
$
$ START OF DUAL ORHTOGONALIZATION OF ADJOINT MODES.
$
INPUTT1   /, , , ,/-1/4
INPUTT1   /MLL,KLL,,,/0/4 $
MPYAD     ZETAI,MLL, /ZEM/+1 $
MPYAD     ZEM,PHIAI, /KOEF/0 $
DIAGONAL  MLL/UNITY/*SQUARE*/0.0 $
SOLVE     KOEF,UNITY/BETA/-1/+1/+2/+2 $
MPYAD     PHIAI,BETA, /OMEGA/0 $
MPYAD     OMEGA,MLL, /MEGM/+1/+1 $
MPYAD     MEGM,ZETAI, /GENMASS/0 $
MPYAD     OMEGA,KLL, /MEGK/+1/+1 $
MPYAD     MEGK,ZETAI, /GENSTIF/0 $
MATPRN    GENMASS,ZETAI,OMEGA,GENSTIF,// $
OUTPUT3   ZETAI,OMEGA,GENMASS,GENSTIF,//0/C,Y,N1=ZZZ/
          C,Y,N2=MEG/C,Y,N3=MMM/C,Y,N4=KKK $
END $ FINISH OF ORTHOGONALIZATION OF RITZ VECTORS
```

APPENDIX D

DEMONSTRATION PROBLEM FOR UNSYMMETRIC RITZ

RITZ DEMO


Sensor
5-3
SIL=27
Actuator
11-3
SIL=63
OMIT
OMIT
Y-OFF
Y-OFF
Z-OFF
Z-OFF
MPC
Z
X
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15


1,Z
CBAR
2,Y
A=2.0 I1 = 1.0
I2=2.0 J=3.0

LOAD#	G.P.	Direction
1	14	Neg. Axial
2	14	Neg. Trnsv. Y
3	7	Pos. Trnsv. Z
4	14	Pos. Torque

APPENDIX E

```
        GENMASS  KAPPA = .007

        1.00000E+00 -1.57700E-21 -1.75392E-21 -4.23773E-26
        7.33943E-25  1.00000E+00  5.52286E-26  1.84410E-23
        4.86876E-22 -1.76725E-24  1.00000E+00 -1.41962E-23
        6.08346E-27 -4.10067E-23  6.50197E-24  1.00000E+00


        GENMASS  KAPPA = 0.95

  1.00000E+00 -3.41619E-17 -2.08085E-17 -7.77767E-17 -3.55682E-15
        6.80886E-17  7.19268E-16 -3.59782E-16  1.85463E-15
 -2.09800E-17  1.00000E+00 -2.66490E-16  5.34076E-15  4.56150E-17
       -1.0807            2732E-14  1.36521E-14 -1.98969E-13
 -7.27244E-16  3.14776E-15  1.00000E+00  1.58474E-15  6.56536E-16
       -7.71337E-15 -3.17466E-14  1.96339E-14 -4.29536E-14
 -6.40719E-18 -1.66050E-15  5.80912E-17  1.00000E+00  4.28422E-19
        1.45089E-15  7.28284E-15 -5.87202E-15  1.86094E-14
 -1.15957E-15  1.05575E-16  4.21965E-17  1.36209E-16  1.00000E+00
       -2.06182E-16 -1.70354E-15  4.34321E-16 -2.34621E-15
 -4.61736E-17  4.20176E-15 -3.42951E-16  1.05352E-14  5.74604E-17
        1.00000E+00  2.34472E-14  1.11612E-14 -1.95622E-13
 -1.94247E-18 -4.76531E-16  3.61424E-17 -7.44025E-16  4.46676E-18
        1.22754E-15  1.00000E+00 -1.59921E-15  2.45933E-14
  2.42787E-18  8.22934E-16 -2.48630E-17  1.54808E-15 -1.46353E-17
       -6.75527E-16 -7.19174E-16  1.00000E+00 -1.18324E-14
  1.14403E-18 -1.60473E-17 -1.99198E-18  6.85738E-18 -1.30097E-18
        3.33071E-17 -6.84577E-18 -9.50914E-17  1.00000E+00
```

BEST COPY
AVAILABLE

```
GENSTIF  KAPPA = .007

 1.04682E+01  6.15654E-04 -9.26258E-04 -2.98338E-10
-3.08635E-04  4.61758E-02  2.74302E-08  5.87860E-08
-8.90200E-04 -5.23407E-08  1.36253E-02 -3.94594E-08
-8.76229E-10  2.54442E-05 -1.91582E-05  2.57602E+01
```

```
        GENSTIF  KAPPA = 0.95

 3.39196E+02 -6.69263E-01 -2.44002E-01 -9.04235E-01 -3.12387E+02
        1.17373E+00  2.61338E+01 -1.37057E+00  2.05697E+01
-1.08090E-02  3.15758E-01 -5.48427E-03  1.31833E-01  1.07321E-02
       -1.70708E-01  8.53201E-01  6.03132E-01 -2.68652E+00
-3.09520E-02  1.61339E-02  2.06822E-02  2.84852E-02  2.85713E-02
       -3.69205E-02 -2.37364E-01  1.00253E-01 -6.03664E-01
-8.47928E-01  1.11503E+02 -8.23610E+00  6.49374E+02  1.19001E+00
       -8.02691E+02  3.80773E+02  3.22937E+02 -4.57428E+03
-3.07444E+02  6.23311E-01  2.25336E-01  8.38877E-01  2.89818E+02
       -1.08889E+00 -2.42260E+01  1.27283E+00 -1.90819E+01
 7.98571E-01 -1.04804E+02  7.74130E+00 -5.92276E+02 -1.12001E+00
        7.54470E+02 -3.57896E+02 -3.03535E+02  4.29948E+03
 1.98390E-01  2.15040E-02 -8.83532E-03  2.23315E-02 -1.85797E-01
       -2.80937E-02  3.86155E-01 -1.58374E-02  6.00169E-02
 6.36402E-02  1.95278E+00 -1.21436E-01  4.85570E+00 -5.51377E-02
       -6.20209E+00  6.10810E+00  5.60222E+00 -7.46195E+01
 1.76325E-02 -2.24893E+00  1.66211E-01 -5.93498E+00 -2.44872E-02
        7.57854E+00 -7.62053E+00 -6.48352E+00  9.22406E+01
```

MODAL STRAIN ENERGIES IN COSMIC NASTRAN

B. D. Snyder and V. B. Venkayya
Flight Dynamics Laboratory
Wright Patterson AFB

SUMMARY

A computer program was developed to take a NASTRAN output file from a normal modes analysis and calculate the modal strain energies of selected elements. The FORTRAN program can determine the modal strain energies for CROD, CBAR, CELAS, CTRMEM, CQDMEM2, and CSHEAR elements. Modal strain energies are useful in estimating damping in structures.

INTRODUCTION

This work was initiated to predict damping in a large passively damped truss structure. The twelve meter truss structure is currently undergoing modal testing in preparation for controls experiments. An estimate of the total damping in the structure is needed for the controls experiments.

The starting point for the computer program was the ANALYZE program (ref. 1). First, the information needed for the strain energies was extracted from a NASTRAN output file (ref. 2). Element information is extracted from the echo of the bulk data and eigenvector information is extracted from the eigenvector tables for each mode. The element stiffness matrices are formed and then multiplied by the appropriate element eigenvector and its transpose and divided by two. The result is the element strain energies for a given mode. With this information, the modal strain energy method can be used to predict the damping in a viscoelastically damped structure. COSMIC NASTRAN can output element strain energies but only for static analyses. To predict the structural damping, the element modal strain energies for a normal modes analysis have to be found.

SYMBOLS

w_x = x displacements in the plane of the plate in the local coordinate system

w_y = y displacements in the plane of the plate in the local coordinate system

$a_1, b_1, c_1, a_2, b_2, c_2$ = six undetermined coefficients

$x_1, y_1, \ldots, x_3, y_3$ = coordinates of the 3 nodes of the triangle in the local coordinate system

$\underset{\sim}{\eta}$ = shape matrix

$\underset{\sim}{\sigma}$ = stress vector

$\underset{\sim}{\epsilon}$ = strain vector

G = shear modulus

E = modulus of elasticity

$\underset{\sim}{k}$ = element stiffness matrix

$\underset{\sim}{\phi}^r$ = element eigenvector for the rth mode

ELEMENT FORMULATION

As mentioned previously, the formulation of the elements comes from the ANALYZE program. The element stiffness matrices are exactly the same as the COSMIC NASTRAN formulation for the CELAS, CBAR, and CROD elements. The formulation for the CTRMEM and CQDMEM2 elements is slightly different than COSMIC NASTRAN. The CSHEAR formulation is very different from the COSMIC NASTRAN formulation. The basis for the derivation of the shear panel is empirical but accurately constructed finite element models produce satisfactory results. The modal strain energy program will produce good results if the shear panel planform is as close to rectangular as possible. The less skewing of the element, the better the results will be.

The triangular membrane element used in this program is a constant strain plate element. The quadrilateral membrane and shear elements are constructed of four (non-overlapping) of the constant strain triangular membrane elements mentioned above. The elements are assumed to be flat plates which means the warping in the elements is ignored. The elements have a fictitious interior node which is later removed by static condensation. Only shear energy is considered in the stiffness of the shear element where the quadrilateral membrane element considers all the energy in the element.

Since the triangular membrane element is the basis for all the other plate elements in this program, the derivation will be given along with how these triangle elements are used to formulate the quadrilateral and shear elements. The linear displacement field in the triangular element can be represented by

$$w_x = a_1 x + b_1 y + c_1 \tag{1}$$

$$w_y = a_2 x + b_2 y + c_2 \tag{2}$$

or in matrix form

$$\underset{\sim}{w} = \begin{pmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ b_1 \\ c_1 \\ a_2 \\ b_2 \\ c_2 \end{pmatrix} \tag{3}$$

The six unknown coefficients can be uniquely determined by the six boundary conditions at the nodes.

$$\left(\begin{array}{c} v_1 \\ v_3 \\ v_5 \\ \hline v_2 \\ v_4 \\ v_6 \end{array}\right) = \left(\begin{array}{ccc|ccc} x_1 & y_1 & 1 & 0 & 0 & 0 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & x_1 & y_1 & 1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \end{array}\right) \left(\begin{array}{c} a_1 \\ b_1 \\ c_1 \\ \hline a_2 \\ b_2 \\ c_2 \end{array}\right) \tag{4}$$

The inversion of the partitioned diagonal matrix involves simply the inversion of the component matrix. The shape matrix $\underset{\sim}{\eta}$ is given by

$$\underset{\sim}{\eta} = \underset{\sim}{x}\, \underset{\sim}{Z}^{-1} \tag{5}$$

where the matrix $\underset{\sim}{x}$ is given by

$$\underset{\sim}{x} = \begin{pmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{pmatrix} \tag{6}$$

and the $\underset{\sim}{Z}$ matrix is given by

$$\underset{\sim}{Z} = \left(\begin{array}{c|c} \underset{\sim}{X} & 0 \\ \hline 0 & \underset{\sim}{X} \end{array}\right) \tag{7}$$

The coordinate matrix $\underset{\sim}{X}$ is given by

$$\underset{\sim}{X} = \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{pmatrix} \tag{8}$$

From linear strain-displacement relations, the strains can be written as

$$\epsilon_x = \frac{\partial w_x}{\partial x} = a_1 \tag{9}$$

$$\epsilon_y = \frac{\partial w_y}{\partial y} = b_2 \tag{10}$$

$$\epsilon_{xy} = \frac{\partial w_x}{\partial y} + \frac{\partial w_y}{\partial x} = b_1 + a_2 \tag{11}$$

From the principle of virtual work, the elements of the member stiffness matrix can be written as

$$k_{ij} = \int_V \underset{\sim}{\sigma}^{(i)^t} \underset{\sim}{\epsilon}^{(j)}\, dV = \int_V \underset{\sim}{\epsilon}^{(i)^t}\, \underset{\sim}{E}\, \underset{\sim}{\epsilon}^{(j)}\, dV \tag{12}$$

where $\underset{\sim}{\sigma}^{(i)}$ and $\underset{\sim}{\epsilon}^{(j)}$ are the stress and strain matrices corresponding to the unit displacement modes explained in equation 8. Since the linear displacement relation implies constant strain, the integral in equation 12 can be replaced by the volume of the element:

$$k_{ij} = \frac{1}{2}\, |\underset{\sim}{X}|\, t\, \underset{\sim}{\epsilon}^{(i)^t} \underset{\sim}{E}\, \underset{\sim}{\epsilon}^{(j)} \tag{13}$$

where $|\underset{\sim}{X}|$ is the determinant of the nodal coordinate matrix which represents twice the area of the element and t is the thickness of the element. Finally, the stiffness matrix of the triangular membrane element is given by

$$\underset{\sim}{k} = \frac{1}{2}\,|\underset{\sim}{X}|\,t \begin{pmatrix} \underset{\sim}{\epsilon}^{(1)^t} & \underset{\sim}{E}\underset{\sim}{\epsilon}^{(1)} & \underset{\sim}{\epsilon}^{(1)^t} & \underset{\sim}{E}\underset{\sim}{\epsilon}^{(2)} & \cdots & \underset{\sim}{\epsilon}^{(1)^t} & \underset{\sim}{E}\underset{\sim}{\epsilon}^{(6)} \\ \underset{\sim}{\epsilon}^{(2)^t} & \underset{\sim}{E}\underset{\sim}{\epsilon}^{(1)} & \underset{\sim}{\epsilon}^{(2)^t} & \underset{\sim}{E}\underset{\sim}{\epsilon}^{(2)} & \cdots & \underset{\sim}{\epsilon}^{(2)^t} & \underset{\sim}{E}\underset{\sim}{\epsilon}^{(6)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ \underset{\sim}{\epsilon}^{(6)^t} & \underset{\sim}{E}\underset{\sim}{\epsilon}^{(1)} & \underset{\sim}{\epsilon}^{(6)^t} & \underset{\sim}{E}\underset{\sim}{\epsilon}^{(2)} & \cdots & \underset{\sim}{\epsilon}^{(6)^t} & \underset{\sim}{E}\underset{\sim}{\epsilon}^{(6)} \end{pmatrix} \tag{14}$$

Equation 14 gives the formulation for the stiffness matrix of the triangular membrane elements. What follows, is how four of these triangular elements are used to construct quadrilateral membrane and shear elements.

The stiffness matrix of the quadrilateral membrane element is determined by breaking it into four component triangles. The fictitious node in the quadrilateral is located by averaging the coordinates of the four nodes as follows

$$x_5 = \frac{x_1 + x_2 + x_3 + x_4}{4} \tag{15}$$

$$y_5 = \frac{y_1 + y_2 + y_3 + y_4}{4} \tag{16}$$

The stiffness of the four triangles can then be computed by equation 14. Addition of the four stiffness matrices gives a 10 × 10 stiffness matrix with two degrees of freedom included for the fifth node. The force displacement relations of the five node quadrilateral are written as

$$\underset{\sim}{R}_Q = \underset{\sim}{k}_Q\,\underset{\sim}{r}_Q \tag{17}$$

where the subscript refers to the quadrilateral element with five nodes. Equation 17, partitioned to isolate the degrees of freedom of the fifth node can be written as

$$\begin{pmatrix} \underset{\sim}{R}_I \\ \hline \underset{\sim}{R}_{II} \end{pmatrix} = \left(\begin{array}{c|c} \underset{\sim}{k}_{I,I} & \underset{\sim}{k}_{I,II} \\ \hline \underset{\sim}{k}_{II,I} & \underset{\sim}{k}_{II,II} \end{array}\right) \begin{pmatrix} \underset{\sim}{r}_I \\ \hline \underset{\sim}{r}_{II} \end{pmatrix} \tag{18}$$

Equation 18 can be written as two separate equations

$$\underset{\sim}{R}_I = \underset{\sim}{k}_{I,I}\,\underset{\sim}{r}_I + \underset{\sim}{k}_{I,II}\,\underset{\sim}{r}_{II} \tag{19}$$

$$\underset{\sim}{R}_{II} = \underset{\sim}{k}_{II,I}\,\underset{\sim}{r}_I + \underset{\sim}{k}_{II,II}\,\underset{\sim}{r}_{II} \tag{20}$$

Since the fifth node doesn't actually exist in the original model, no external forces can be applied to this node. This condition gives

$$\underset{\sim}{r}_{II} = -\underset{\sim}{k}^{-1}_{II,II}\,\underset{\sim}{k}_{II,I}\,\underset{\sim}{r}_I \tag{21}$$

Substitution of equation 21 in equation 19 gives

$$\underset{\sim}{R}_I = \left(\underset{\sim}{k}_{I,I} - \underset{\sim}{k}_{I,II}\,\underset{\sim}{k}^{-1}_{II,II}\,\underset{\sim}{k}_{II,I}\right)\underset{\sim}{r}_I \tag{22}$$

From equation 22 the stiffness matrix of the original quadrilateral membrane element can be written as

$$\underline{k} = \underline{k}_{\text{II}} - \underline{k}_{\text{I,II}}\, \underline{k}_{\text{II,II}}^{-1}\, \underline{k}_{\text{II,I}} \tag{23}$$

The shear element is also composed of four triangular elements however, the stiffness matrices of the component triangles are determined by considering only the shear strain energy (equation 13).

$$k_{ij} = \frac{1}{2}\, |\underline{X}|\, t\, \epsilon_{xy}^{(i)} G \epsilon_{xy}^{(j)} \tag{24}$$

MODAL STRAIN ENERGY PROGRAM

The program starts by reading in all the information it needs from a NASTRAN output file for a normal modes analysis. As the program is set up, it can handle 1,000 of any one type of element for a total of 6,000 elements. A total of 100 materials can be specified but only isotropic materials specified on MAT1 cards are currently accounted for. The model can have 100 properties for any one element type for a total of 600 property cards. These limits can easily be expanded by changing the dimensions of the arrays in the code. The CELAS elements (CELAS1 or CELAS2) must be grounded (fixed) at one end with the other end connected to the structure. There are two ways to do this, one is to leave the second grid point of the CELAS card and its component blank or the second way is to specify a second grid point and component and then fix the second grid point component with an SPC card.

The next step in determining the modal strain energies is to calculate the element stiffness matrices. Using the equations derived above and equations for the CELAS, CBAR, and CROD elements, the stiffness matrices are generated. After this the eigenvector for the current element is extracted from the eigenvector table for a given mode. Then the following equation is used to determine the element modal strain energies for the given mode

$$\textit{Element Modal Strain Energy} = \frac{1}{2}\underline{\phi}^{r^t}\, \underline{k}\, \underline{\phi}^{r} \tag{25}$$

The equation is used for each element for every mode printed in the NASTRAN normal modes analysis.

After the element strain energies are calculated, they are printed in an easy to read format. The modal strain energy program prints out the following quantities for each mode: element ID number (EID), element type (CBAR, CELAS, CROD, CTRMEM, CSHEAR, or CQDMEM2), element strain energy (in consistent units), percent element strain energy of the entire structure, sum of the total element strain energy for each element type, and the total element strain energy for the entire structure. The program also prints one-half the generalized stiffness from the NASTRAN output file as a check. One-half the generalized stiffness should equal the total strain energy for the entire structure.

APPLICATIONS

Viscoelastic materials are seeing widespread use to suppress vibrations in all types of structures. The ability of viscoelastic materials to passively damp vibrations in lightweight structures is well documented. Modal strain energies are useful in estimating the damping in this type of structure. The approach used to predict the modal damping (loss) factors for each mode of the structure is called the modal strain energy method. It states that the ratio of structural loss factor to viscoelastic material loss factor for a given mode of vibration can be estimated as the ratio of elastic strain energy in the viscoelastic to total elastic strain energy in the entire structure when it deforms into the particular undamped mode shape (ref. 3). Mathematically this can be stated as

$$\frac{\eta_s^{(r)}}{\eta_v} = \frac{V_v^{(r)}}{V_s^{(r)}} \tag{26}$$

where

η_s^r = loss factor for the r'th mode of the composite structure

η_v = material loss factor for the viscoelastic material

V_v^r = elastic strain energy stored in the viscoelastic material when the structure deforms in its r'th undamped mode shape

V_s^r = elastic strain energy of the entire composite structure in the r'th mode shape

Computing the undamped mode shapes of the composite structure with the viscoelastic material treated as if it were purely elastic with a real stiffness modulus, the right hand side of equation 26 is calculated as

$$\frac{V_v^r}{V_s^r} = \frac{\sum_{\theta=1}^{n} \underset{\sim}{\phi}_\theta^{r^t} \underset{\sim}{k}_\theta \underset{\sim}{\phi}_\theta^r}{\underset{\sim}{\phi}^{r^t} \underset{\sim}{K} \underset{\sim}{\phi}^r} \tag{27}$$

where

$\underset{\sim}{\phi}^r$ = r'th mode shape vector

$\underset{\sim}{\phi}_\theta^r$ = subvector formed by deleting from $\underset{\sim}{\phi}$ all entries not corresponding to motion of nodes of the θ'th viscoelastic element

$\underset{\sim}{k}_\theta$ = element stiffness matrix of the θ'th viscoelastic element

$\underset{\sim}{K}$ = stiffness matrix of the entire composite structure

n = number of viscoelastic elements in the model

Combining equations 26 and 27 you get (ref. 4)

$$\eta_s^r = \frac{\sum_{\theta=1}^{n} \eta_{v_\theta} \underset{\sim}{\phi}_\theta^{r^t} \underset{\sim}{k}_\theta \underset{\sim}{\phi}_\theta^r}{\underset{\sim}{\phi}^{r^t} \underset{\sim}{K} \underset{\sim}{\phi}^r} \tag{28}$$

This equation states that if you create a NASTRAN model of the damped structure with all elements included except damper elements, and then run a normal modes analysis, you have all the information needed to get the structural loss factor. After you make the NASTRAN run, you run the output through the element modal strain energy program which gives you the percentages of element strain energy to total strain energy for the entire structure. The percentages for the elements that actually possess viscoelastic damping are multiplied by that particular elements material loss factor. These quantities are then summed to give the loss factor for the entire structure.

EXAMPLE PROBLEMS

Three example problems were used to demonstrate the ability of the element strain energy program to accurately output the element modal strain energies. The first problem is a rectangular wing box and it is shown broken up into its numbered elements in figure 1. The rectangular wing box consists of quadrilateral membrane elements for the inboard top and bottom skins, triangular membrane elements for the outboard top and bottom skins, bar elements for the outboard posts, rod elements for all other posts, shear elements for all the ribs and spars, elastic elements provide the inboard top skin attachment points with the inboard bottom skin points rigidly fixed. The strain energy outputs for the second mode of this model are given in table I. This model contains all the element types the strain energy program is capable of handling. Comparing the total structural strain energy with the value for one-half the generalized stiffness shows that the two are in agreement.

The second example is known as the intermediate complexity wing and is just a simplified NASTRAN model of the load carrying portion of a wing. Shown in figure 2 broken into its component elements and their numbering scheme, is a depiction of the wing. The model consists of quadrilateral membrane elements for most of the top and bottom skins, two triangular membrane elements for the outboard corner elements on the top and bottom skins, rod elements for all the posts, and shear elements for all the ribs and spars. The inboard top and bottom skin points rigidly fixed. The strain energy program output for the first mode of the model is given in table II. As you can see, the program accurately produces zero strain energy in all the rod elements for the first bending mode of the wing. The difference in the values for the total structural strain energy and one-half the generalized stiffness can be attributed to the different formulations of the stiffness matrices of the plate membrane and shear elements.

The final example is a part of the Large Space Structures Technology Program at the Flight Dynamics Laboratory. A NASTRAN model of the twelve meter truss structure is shown in figure 3. The elements aren't numbered because of the large number of elements in the model. The model consists of bar elements for the horizontal and vertical elements and rod elements for all the diagonals. The diagonal members contain the viscoelastic dampers on the actual structure. The model is supported at the base with a series of elastic elements. The strain energy output for the second mode is given in table III. This is the first torsion mode of the truss, so most of the strain energy is in the diagonal members. This is verified by the modal strain energy program. The loss factor for the entire structure has been predicted and is awaiting test results for verification.

CONCLUDING REMARKS

A FORTRAN program that calculates element strain energies has been developed and verified. This program gives COSMIC NASTRAN a capability that was only previously available for static analysis. Work is currently underway to develop DMAP instructions to calculate modal strain energies directly in NASTRAN. With the ever increasing trend toward lighter structures, damping materials will see increased use in all types of structures. A simple, accurate method, such as the modal strain energy method, to predict structural damping is essential.

REFERENCES

1. Venkayya, V.B. and Tischler, V.A., ""ANALYZE" - Analysis of Aerospace Structures with Membrane Elements", AFFDL-TR-78-170, Aug. 1978.

2. The NASTRAN User's Manual, Jun. 1985.

3. Johnson, C.D., Kienholz, D.A., and Rogers, L.C., "Finite Element Prediction of Damping in Beams with Constrained Viscoelastic Layers," Shock and Vibration Bulletin, No. 50, Part 1, May 1981, pp. 71-82.

4. Johnson, C.D. and Kienholz, D.A., "Design and Testing of a Sixty Foot Damped Generic Space Truss."

TABLE I - RECTANGULAR WING BOX STRAIN ENERGIES

MODE 2

Element ID	Element Type	Element Strain Energy	% Strain Energy
50001	CBAR	-0.4068E-06	0.0000
50002	CBAR	6.446	0.2810
50003	CBAR	-0.6010E-06	0.0000
500	CELAS	0.2516	0.0110
502	CELAS	0.2749E-01	0.0012
503	CELAS	0.2810E-01	0.0012
504	CELAS	69.68	3.0378
505	CELAS	55.31	2.4114
506	CELAS	50.50	2.2015
507	CELAS	0.5442	0.0237
508	CELAS	0.9635E-01	0.0042
509	CELAS	0.1016	0.0044
50004	CROD	0.0000E+00	0.0000
50005	CROD	0.0000E+00	0.0000
50006	CROD	0.0000E+00	0.0000
50007	CROD	1.088	0.0475
50008	CROD	0.1927	0.0084
50009	CROD	0.2032	0.0089
10001	CTRMEM	117.4	5.1190
10002	CTRMEM	116.7	5.0897

10011	CTRMEM	137.4	5.9880
10012	CTRMEM	117.8	5.1372
20001	CTRMEM	115.1	5.0171
20002	CTRMEM	119.7	5.2185
20011	CTRMEM	132.1	5.7594
20012	CTRMEM	115.9	5.0520
10003	CQDMEM	218.9	9.5449
10004	CQDMEM	166.3	7.2488
20003	CQDMEM	200.6	8.7462
20004	CQDMEM	189.0	8.2413
30001	CSHEAR	29.60	1.2904
30002	CSHEAR	88.67	3.8657
30003	CSHEAR	34.38	1.4987
30004	CSHEAR	80.61	3.5140
30005	CSHEAR	72.90	3.1781
30006	CSHEAR	28.34	1.2355
40001	CSHEAR	12.29	0.5360
40002	CSHEAR	8.467	0.3691
40003	CSHEAR	0.6263E-02	0.0003
40004	CSHEAR	3.315	0.1445
40005	CSHEAR	3.752	0.1636
40006	CSHEAR	0.9803E-05	0.0000

STRAIN ENERGY IN CELAS ELEMENTS = 176.5

STRAIN ENERGY IN CBAR ELEMENTS = 6.446

STRAIN ENERGY IN CROD ELEMENTS = 1.484

STRAIN ENERGY IN CTRMEM ELEMENTS = 972.1

STRAIN ENERGY IN CQDMEM ELEMENTS = 774.9

STRAIN ENERGY IN CSHEAR ELEMENTS = 362.3

TOTAL STRAIN ENERGY = 2294.

GENERALIZED STIFFNESS/2 = 2294.

TABLE II - INTERMEDIATE COMPLEXITY WING STRAIN ENERGIES

MODE 1

Element ID	Element Type	Element Strain Energy	% Strain Energy
120	CROD	0.0000E+00	0.0000
121	CROD	0.0000E+00	0.0000
122	CROD	0.0000E+00	0.0000
123	CROD	0.0000E+00	0.0000
124	CROD	0.0000E+00	0.0000
125	CROD	0.0000E+00	0.0000
126	CROD	0.0000E+00	0.0000
127	CROD	0.0000E+00	0.0000
128	CROD	0.0000E+00	0.0000

129	CROD	0.0000E+00	0.0000
130	CROD	0.0000E+00	0.0000
131	CROD	0.0000E+00	0.0000
132	CROD	0.0000E+00	0.0000
133	CROD	0.0000E+00	0.0000
134	CROD	0.0000E+00	0.0000
135	CROD	0.0000E+00	0.0000
136	CROD	0.0000E+00	0.0000
137	CROD	0.0000E+00	0.0000
138	CROD	0.0000E+00	0.0000
139	CROD	0.0000E+00	0.0000
140	CROD	0.0000E+00	0.0000
141	CROD	0.0000E+00	0.0000
142	CROD	0.0000E+00	0.0000
143	CROD	0.0000E+00	0.0000
144	CROD	0.0000E+00	0.0000
145	CROD	0.0000E+00	0.0000
146	CROD	0.0000E+00	0.0000
147	CROD	0.0000E+00	0.0000
148	CROD	0.0000E+00	0.0000
149	CROD	0.0000E+00	0.0000
150	CROD	0.0000E+00	0.0000
151	CROD	0.0000E+00	0.0000
152	CROD	0.0000E+00	0.0000
153	CROD	0.0000E+00	0.0000
154	CROD	0.0000E+00	0.0000
155	CROD	0.0000E+00	0.0000
156	CROD	0.0000E+00	0.0000
157	CROD	0.0000E+00	0.0000
158	CROD	0.0000E+00	0.0000
1	CTRMEM	0.2817E-01	0.0046
2	CTRMEM	0.2817E-01	0.0046
3	CQDMEM	0.8132E-01	0.0131
4	CQDMEM	0.8132E-01	0.0131
5	CQDMEM	0.6809E-01	0.0110
6	CQDMEM	0.6809E-01	0.0110
7	CQDMEM	0.1561	0.0252
8	CQDMEM	0.1561	0.0252
9	CQDMEM	0.6461	0.1045
10	CQDMEM	0.6461	0.1045
11	CQDMEM	0.7805	0.1262
12	CQDMEM	0.7805	0.1262
13	CQDMEM	0.7629	0.1234
14	CQDMEM	0.7629	0.1234
15	CQDMEM	0.9026	0.1459
16	CQDMEM	0.9026	0.1459
17	CQDMEM	2.573	0.4161
18	CQDMEM	2.573	0.4161
19	CQDMEM	3.169	0.5124
20	CQDMEM	3.169	0.5124
21	CQDMEM	3.299	0.5335
22	CQDMEM	3.299	0.5335
23	CQDMEM	3.306	0.5345

24	CQDMEM	3.306	0.5345
25	CQDMEM	5.762	0.9316
26	CQDMEM	5.762	0.9316
27	CQDMEM	7.520	1.2159
28	CQDMEM	7.520	1.2159
29	CQDMEM	7.967	1.2882
30	CQDMEM	7.967	1.2882
31	CQDMEM	7.312	1.1824
32	CQDMEM	7.312	1.1824
33	CQDMEM	9.504	1.5367
34	CQDMEM	9.504	1.5367
35	CQDMEM	12.89	2.0840
36	CQDMEM	12.89	2.0840
37	CQDMEM	14.20	2.2956
38	CQDMEM	14.20	2.2956
39	CQDMEM	12.59	2.0364
40	CQDMEM	12.59	2.0364
41	CQDMEM	12.67	2.0479
42	CQDMEM	12.67	2.0479
43	CQDMEM	17.60	2.8462
44	CQDMEM	17.60	2.8462
45	CQDMEM	21.24	3.4346
46	CQDMEM	21.24	3.4346
47	CQDMEM	19.38	3.1329
48	CQDMEM	19.38	3.1329
49	CQDMEM	13.58	2.1953
50	CQDMEM	13.58	2.1953
51	CQDMEM	17.25	2.7891
52	CQDMEM	17.25	2.7891
53	CQDMEM	20.66	3.3399
54	CQDMEM	20.66	3.3399
55	CQDMEM	16.06	2.5971
56	CQDMEM	16.06	2.5971
57	CQDMEM	6.645	1.0744
58	CQDMEM	6.645	1.0744
59	CQDMEM	17.18	2.7780
60	CQDMEM	17.18	2.7780
61	CQDMEM	21.31	3.4457
62	CQDMEM	21.31	3.4457
63	CQDMEM	23.16	3.7449
64	CQDMEM	23.16	3.7449
65	CSHEAR	0.4318E-01	0.0070
66	CSHEAR	0.5833E-04	0.0000
67	CSHEAR	0.4419E-01	0.0071
68	CSHEAR	0.6681E-03	0.0001
69	CSHEAR	0.1559E-02	0.0003
70	CSHEAR	0.3317E-01	0.0054
71	CSHEAR	0.3689E-02	0.0006
72	CSHEAR	0.1579E-01	0.0026
73	CSHEAR	0.3236E-02	0.0005
74	CSHEAR	0.6835E-02	0.0011
75	CSHEAR	0.3295E-02	0.0005
76	CSHEAR	0.7111E-02	0.0011

77	CSHEAR	0.3093E-02	0.0005
78	CSHEAR	0.4573E-02	0.0007
79	CSHEAR	0.2292E-02	0.0004
80	CSHEAR	0.4423E-02	0.0007
81	CSHEAR	0.3590E-02	0.0006
82	CSHEAR	0.4746E-02	0.0008
83	CSHEAR	0.1902E-02	0.0003
84	CSHEAR	0.4688E-02	0.0008
85	CSHEAR	0.7538E-02	0.0012
86	CSHEAR	0.8932E-02	0.0014
87	CSHEAR	0.3638E-02	0.0006
88	CSHEAR	0.9771E-02	0.0016
89	CSHEAR	0.9090E-01	0.0147
90	CSHEAR	0.6178E-01	0.0100
91	CSHEAR	0.4218E-01	0.0068
92	CSHEAR	0.6935E-01	0.0112
93	CSHEAR	0.5735	0.0927
94	CSHEAR	0.3672	0.0594
95	CSHEAR	0.1479	0.0239
96	CSHEAR	0.2381	0.0385
97	CSHEAR	0.2449	0.0396
98	CSHEAR	0.4526	0.0732
99	CSHEAR	0.5576	0.0902
100	CSHEAR	0.5448	0.0881
101	CSHEAR	0.3861	0.0624
102	CSHEAR	0.8999E-01	0.0146
103	CSHEAR	0.7155	0.1157
104	CSHEAR	0.1065	0.0172
105	CSHEAR	0.8149	0.1318
106	CSHEAR	1.047	0.1692
107	CSHEAR	1.125	0.1819
108	CSHEAR	1.076	0.1740
109	CSHEAR	1.026	0.1659
110	CSHEAR	0.9453	0.1528
111	CSHEAR	1.735	0.2805
112	CSHEAR	0.5387E-01	0.0087
113	CSHEAR	0.3877	0.0627
114	CSHEAR	0.6420	0.1038
115	CSHEAR	0.7546	0.1220
116	CSHEAR	0.7662	0.1239
117	CSHEAR	0.7926	0.1281
118	CSHEAR	0.5955	0.0963
119	CSHEAR	1.297	0.2097

STRAIN ENERGY IN CROD ELEMENTS = 0.0000E+00

STRAIN ENERGY IN CTRMEM ELEMENTS = 0.5633E-01

STRAIN ENERGY IN CQDMEM ELEMENTS = 600.4

STRAIN ENERGY IN CSHEAR ELEMENTS = 17.97

TOTAL STRAIN ENERGY = 618.5

GENERALIZED STIFFNESS/2 = 619.0

TABLE III - TWELVE METER TRUSS STRAIN ENERGIES

MODE 2

Element ID	Element Type	Element Strain Energy	% Strain Energy
101	CBAR	1.740	6.0085
102	CBAR	1.724	5.9537
103	CBAR	1.127	3.8917
104	CBAR	1.155	3.9862
106	CBAR	0.6792	2.3449
107	CBAR	0.6611	2.2827
108	CBAR	0.3441	1.1880
109	CBAR	0.3522	1.2160
111	CBAR	0.1476	0.5098
112	CBAR	0.1436	0.4956
113	CBAR	0.4434E-01	0.1531
114	CBAR	0.4531E-01	0.1564
116	CBAR	0.7187E-02	0.0248
117	CBAR	0.6948E-02	0.0240
118	CBAR	0.2278E-03	0.0008
119	CBAR	0.2049E-03	0.0007
120	CBAR	0.1767E-01	0.0610
121	CBAR	0.5047E-02	0.0174
122	CBAR	0.4326E-02	0.0149
123	CBAR	0.5652E-02	0.0195
125	CBAR	0.2255E-02	0.0078
126	CBAR	0.3096E-02	0.0107
127	CBAR	0.1182E-02	0.0041
128	CBAR	0.1833E-02	0.0063
130	CBAR	0.3633E-03	0.0013
131	CBAR	0.6576E-03	0.0023
132	CBAR	0.1058E-03	0.0004
133	CBAR	0.2597E-03	0.0009
135	CBAR	0.2277E-04	0.0001
136	CBAR	0.3004E-04	0.0001
137	CBAR	0.4012E-04	0.0001
138	CBAR	0.1046E-04	0.0000
139	CBAR	1.740	6.0085
140	CBAR	1.724	5.9537
141	CBAR	1.127	3.8917
142	CBAR	1.155	3.9862
144	CBAR	0.6792	2.3449
145	CBAR	0.6611	2.2827
146	CBAR	0.3441	1.1880
147	CBAR	0.3522	1.2160
149	CBAR	0.1476	0.5098
150	CBAR	0.1436	0.4956
151	CBAR	0.4434E-01	0.1531
152	CBAR	0.4531E-01	0.1564
154	CBAR	0.7187E-02	0.0248

155	CBAR	0.6948E-02	0.0240
156	CBAR	0.2278E-03	0.0008
157	CBAR	0.2049E-03	0.0007
158	CBAR	0.1767E-01	0.0610
159	CBAR	0.5047E-02	0.0174
160	CBAR	0.4326E-02	0.0149
161	CBAR	0.5652E-02	0.0195
163	CBAR	0.2255E-02	0.0078
164	CBAR	0.3096E-02	0.0107
165	CBAR	0.1182E-02	0.0041
166	CBAR	0.1833E-02	0.0063
168	CBAR	0.3633E-03	0.0013
169	CBAR	0.6576E-03	0.0023
170	CBAR	0.1058E-03	0.0004
171	CBAR	0.2597E-03	0.0009
173	CBAR	0.2277E-04	0.0001
174	CBAR	0.3004E-04	0.0001
175	CBAR	0.4012E-04	0.0001
176	CBAR	0.1046E-04	0.0000
201	CBAR	0.1054E-03	0.0004
202	CBAR	0.9258E-03	0.0032
203	CBAR	0.6164E-03	0.0021
204	CBAR	0.6752E-03	0.0023
205	CBAR	0.5726E-03	0.0020
206	CBAR	0.5726E-03	0.0020
207	CBAR	0.5583E-03	0.0019
208	CBAR	0.5189E-03	0.0018
209	CBAR	0.4704E-03	0.0016
210	CBAR	0.3950E-03	0.0014
211	CBAR	0.3950E-03	0.0014
212	CBAR	0.3261E-03	0.0011
213	CBAR	0.2753E-03	0.0010
214	CBAR	0.2157E-03	0.0007
215	CBAR	0.1482E-03	0.0005
216	CBAR	0.1482E-03	0.0005
217	CBAR	0.8562E-04	0.0003
218	CBAR	0.4871E-04	0.0002
219	CBAR	0.1600E-04	0.0001
220	CBAR	0.8716E-05	0.0000
221	CBAR	0.1055E-03	0.0004
222	CBAR	0.9257E-03	0.0032
223	CBAR	0.6163E-03	0.0021
224	CBAR	0.6755E-03	0.0023
225	CBAR	0.5725E-03	0.0020
226	CBAR	0.5725E-03	0.0020
227	CBAR	0.5583E-03	0.0019
228	CBAR	0.5191E-03	0.0018
229	CBAR	0.4706E-03	0.0016
230	CBAR	0.3950E-03	0.0014
231	CBAR	0.3950E-03	0.0014
232	CBAR	0.3260E-03	0.0011
233	CBAR	0.2754E-03	0.0010
234	CBAR	0.2158E-03	0.0007

235	CBAR	0.1482E-03	0.0005
236	CBAR	0.1482E-03	0.0005
237	CBAR	0.8558E-04	0.0003
238	CBAR	0.4871E-04	0.0002
239	CBAR	0.1596E-04	0.0001
240	CBAR	0.8653E-05	0.0000
241	CBAR	0.1054E-03	0.0004
242	CBAR	0.9258E-03	0.0032
243	CBAR	0.6164E-03	0.0021
244	CBAR	0.6752E-03	0.0023
245	CBAR	0.5726E-03	0.0020
246	CBAR	0.5726E-03	0.0020
247	CBAR	0.5583E-03	0.0019
248	CBAR	0.5189E-03	0.0018
249	CBAR	0.4704E-03	0.0016
250	CBAR	0.3950E-03	0.0014
251	CBAR	0.3950E-03	0.0014
252	CBAR	0.3261E-03	0.0011
253	CBAR	0.2753E-03	0.0010
254	CBAR	0.2157E-03	0.0007
255	CBAR	0.1482E-03	0.0005
256	CBAR	0.1482E-03	0.0005
257	CBAR	0.8562E-04	0.0003
258	CBAR	0.4871E-04	0.0002
259	CBAR	0.1600E-04	0.0001
260	CBAR	0.8716E-05	0.0000
261	CBAR	0.1055E-03	0.0004
262	CBAR	0.9257E-03	0.0032
263	CBAR	0.6163E-03	0.0021
264	CBAR	0.6755E-03	0.0023
265	CBAR	0.5725E-03	0.0020
266	CBAR	0.5725E-03	0.0020
267	CBAR	0.5583E-03	0.0019
268	CBAR	0.5191E-03	0.0018
269	CBAR	0.4706E-03	0.0016
270	CBAR	0.3950E-03	0.0014
271	CBAR	0.3950E-03	0.0014
272	CBAR	0.3260E-03	0.0011
273	CBAR	0.2754E-03	0.0010
274	CBAR	0.2158E-03	0.0007
275	CBAR	0.1482E-03	0.0005
276	CBAR	0.1482E-03	0.0005
277	CBAR	0.8558E-04	0.0003
278	CBAR	0.4871E-04	0.0002
279	CBAR	0.1596E-04	0.0001
280	CBAR	0.8652E-05	0.0000
301	CBAR	0.7572E-01	0.2614
302	CBAR	0.5778E-01	0.1995
303	CBAR	0.9422E-01	0.3253
304	CBAR	0.5765E-01	0.1990
305	CBAR	0.8604E-01	0.2971
306	CBAR	0.5167E-01	0.1784
307	CBAR	0.6932E-01	0.2393

308	CBAR	0.4699E-01	0.1622
309	CBAR	0.5217E-01	0.1801
310	CBAR	0.3316E-01	0.1145
311	CBAR	0.3317E-01	0.1145
312	CBAR	0.2202E-01	0.0760
313	CBAR	0.1538E-01	0.0531
314	CBAR	0.7785E-02	0.0269
315	CBAR	0.3781E-02	0.0131
316	CBAR	0.8850E-03	0.0031
317	CBAR	0.7571E-01	0.2614
318	CBAR	0.5777E-01	0.1995
319	CBAR	0.9421E-01	0.3253
320	CBAR	0.5764E-01	0.1990
321	CBAR	0.8604E-01	0.2971
322	CBAR	0.5167E-01	0.1784
323	CBAR	0.6931E-01	0.2393
324	CBAR	0.4699E-01	0.1622
325	CBAR	0.5218E-01	0.1801
326	CBAR	0.3316E-01	0.1145
327	CBAR	0.3318E-01	0.1145
328	CBAR	0.2202E-01	0.0760
329	CBAR	0.1539E-01	0.0531
330	CBAR	0.7787E-02	0.0269
331	CBAR	0.3783E-02	0.0131
332	CBAR	0.8851E-03	0.0031
333	CBAR	0.7572E-01	0.2614
334	CBAR	0.5778E-01	0.1995
335	CBAR	0.9422E-01	0.3253
336	CBAR	0.5765E-01	0.1990
337	CBAR	0.8604E-01	0.2971
338	CBAR	0.5167E-01	0.1784
339	CBAR	0.6932E-01	0.2393
340	CBAR	0.4699E-01	0.1622
341	CBAR	0.5217E-01	0.1801
342	CBAR	0.3316E-01	0.1145
343	CBAR	0.3317E-01	0.1145
344	CBAR	0.2202E-01	0.0760
345	CBAR	0.1538E-01	0.0531
346	CBAR	0.7786E-02	0.0269
347	CBAR	0.3781E-02	0.0131
348	CBAR	0.8850E-03	0.0031
349	CBAR	0.7571E-01	0.2614
350	CBAR	0.5777E-01	0.1995
351	CBAR	0.9421E-01	0.3253
352	CBAR	0.5764E-01	0.1990
353	CBAR	0.8604E-01	0.2971
354	CBAR	0.5167E-01	0.1784
355	CBAR	0.6931E-01	0.2393
356	CBAR	0.4699E-01	0.1622
357	CBAR	0.5218E-01	0.1801
358	CBAR	0.3316E-01	0.1145
359	CBAR	0.3318E-01	0.1145
360	CBAR	0.2203E-01	0.0760

361	CBAR	0.1539E-01	0.0531
362	CBAR	0.7787E-02	0.0269
363	CBAR	0.3783E-02	0.0131
364	CBAR	0.8852E-03	0.0031
500	CELAS	0.3367E-03	0.0012
501	CELAS	0.2813E-05	0.0000
502	CELAS	0.3367E-03	0.0012
503	CELAS	0.2813E-05	0.0000
504	CELAS	0.3367E-03	0.0012
505	CELAS	0.2812E-05	0.0000
506	CELAS	0.3367E-03	0.0012
507	CELAS	0.2812E-05	0.0000
508	CELAS	0.1465E-16	0.0000
509	CELAS	0.4949E-10	0.0000
510	CELAS	0.1465E-16	0.0000
511	CELAS	0.4949E-10	0.0000
512	CELAS	4.794	16.5529
513	CELAS	0.7808E-10	0.0000
514	CELAS	4.794	16.5529
515	CELAS	0.7808E-10	0.0000
516	CELAS	0.9110E-02	0.0315
517	CELAS	0.8964E-02	0.0309
518	CELAS	0.9110E-02	0.0315
519	CELAS	0.8964E-02	0.0309
520	CELAS	0.9111E-02	0.0315
521	CELAS	0.8964E-02	0.0310
522	CELAS	0.9111E-02	0.0315
523	CELAS	0.8964E-02	0.0310

STRAIN ENERGY IN CELAS ELEMENTS = 9.662

STRAIN ENERGY IN CBAR ELEMENTS = 19.30

TOTAL STRAIN ENERGY = 28.96

GENERALIZED STIFFNESS/2 = 28.96

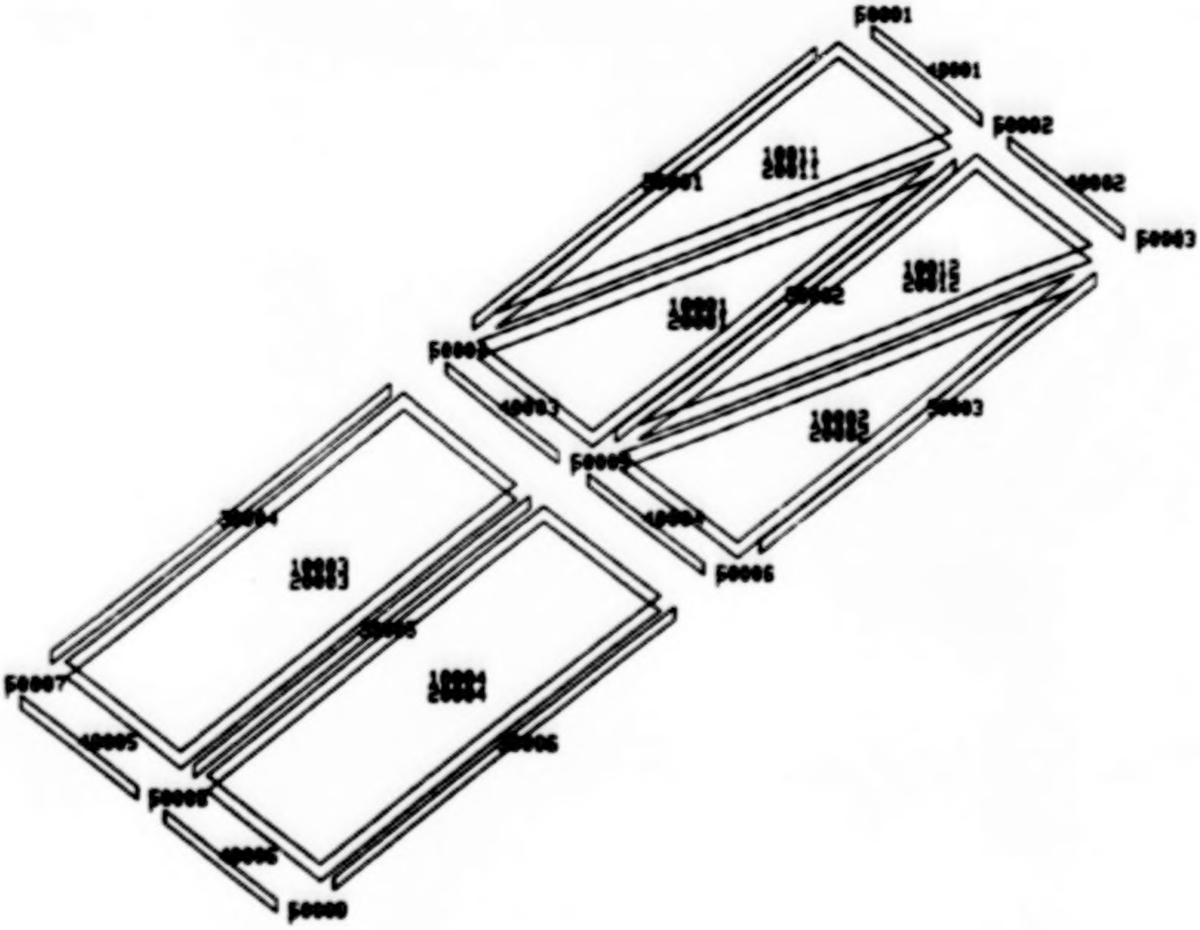

50001
40001
50002
10011
20011
30001
40002
50003
10012
20012
10001
20001
30002
50004
40003
10002
20002
30003
50005
30004
10003
20003
40004
50006
30005
10004
20004
50007
40005
30006
50008
40006
50009

Figure 1 - Rectangular Wing Box Elements

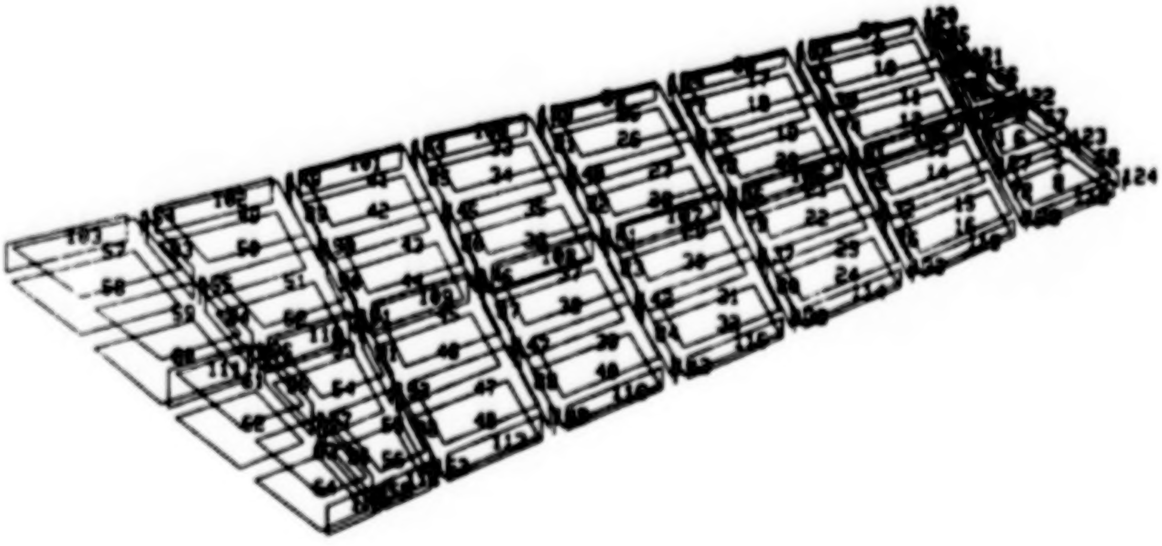

120
124
103
57
58
59

Figure 2 - Intermediate Complexity Wing Elements

BEST COPY
AVAILABLE

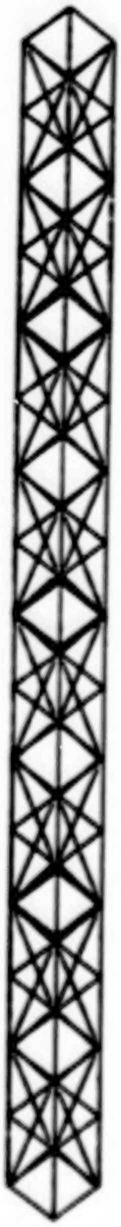


Figure 3 - Twelve Meter Truss Model

Definition of NASTRAN Sets By Use Of Parametric Geometry

by

Terry V Baughn, Associate Professor
Department of Civil & Mechanical Engineering
Southern Methodist University
Dallas, Texas

Mehran Tiv, Graduate Student
Department of Civil & Mechanical Engineering
Southern Methodist University
Dallas, Texas

SUMMARY

Many finite element preprocessors describe finite element model geometry with points, lines, surfaces and volumes. One method for describing these basic geometric entities is by use of parametric cubics which are useful for representing complex shapes. The lines, surfaces and volumes may be discretized for follow on finite element analysis. The ability to limit or selectively recover results from the finite element model is extremely important to the analyst. Equally important is the ability to easily apply boundary conditions. Although graphical preprocessors have made these tasks easier, model complexity may not lend itself to easily identify a group of grid points desired for data recovery or application of constraints. A methodology is presented which makes use of the assignment of grid point locations in parametric coordinates. The parametric coordinates provide a convenient ordering of the grid point locations and a method for retrieving the grid point ID's from the parent geometry. The selected grid points may then be used for the generation of the appropriate set and constraint cards.

LIST OF SYMBOLS

x,y,z	cartesian coordinates
F_1 - F_4	parametric blending functions
L	symbol for a line
R	a vector representing a point in space
$\mathbf{R}_{,\xi}$	derivative of the vector **R**
S	symbol for a surface
$\mathbf{S}_1$ - $\mathbf{S}_4$	vectors that contain algebraic coefficients
V	symbol for a volume
ξ	parametric variable
$(\ )^T$	transpose of a vector

INTRODUCTION

The integration of computer aided drafting programs and finite element preprocessors has led to a common practice of passing basic geometry to the finite element model building programs. The basic geometry may consists of lines, points, arcs and splines which describe the geometry of the part the user wants to analyze. Production finite element pre and post processors make it extremely easy for the structural analyst to make models with a large number of elements and grid points. Generally the input to the preprocessor is kept to a minimum and the program will generate automatically the grid points and elements required to discretize the model. The user may need to identify grid points to apply forces, moments, displacements, temperatures, heat transfer coefficients or select a series of grid points for data recovery. Since the generation is automatic, the user has or little no control of the grid point ID's and will have to rely on the graphics capability of the preprocessor to identify them. The process of selecting the grid point ID's for set or constraint cards can become very time consuming and subject to error if they are manually selected by the user. Most of the advanced model building codes have graphics driven techniques to assist in the selection process. In the absence of a robust preprocessor or if the preprocessor has limited grid selection capability an alternate method may be employed using a neutral file of the model.

Many of the pre and post processors are associated with a specific finite element code. The disc files used for the graphics display of the models can be very large. A common method for reducing the disc space required for storing a finite element model or for transferring the model from one code to another is by use of a neutral file. [1][1] The neutral file is not specific to any code but is a data base of finite element model information. The data base should contain information on grid points, element connectivity, materials, constraints, coordinate systems, etc and model geometry. One of the main objectives of the neutral file is to provide an interface between the pre and post processor and external programs. The model data is written to a neutral file which may be used to extract the desired grid point information. The neutral file is generally readable by the system editor.

The selection of the grid points is based on the relationship between the basic or parent geometry and the finite element data. The basic geometry may then be discretized with the finite element grid points and elements. For example a line may have many grid points placed along its length. The finite element model data base in the form of a neutral file should retain the relationship between the line and the associated grid points. In general the neutral file should contain the relationship between all the geometric data used to generate the model and the associated finite element data. If the basic geometric entities are written in terms of parametric geometry, a procedure may be developed to easily obtain grid point ID's on lines, surfaces and volumes.

PARAMETRIC GEOMETRY

Parametric cubics serve as the mathematical foundation to describe the basic geometric entities such as lines, surfaces and volumes. First consider the development of a parametric curve. A parametric line or curve can be generated from

$$\mathbf{R} = \mathbf{S}_1\xi^3 + \mathbf{S}_2\xi^2 + \mathbf{S}_3\xi + \mathbf{S}_4 \qquad (1)$$

where **R** represents a point in space (x,y & z) and ξ is the parametric variable ranging from zero to one. The vectors **R** and **S** are indicated by bold type. The four vectors, $\mathbf{S}_1$ through $\mathbf{S}_4$ each have three components and may be defined in terms of the end conditions of the line. A parametric curve in a cartesian coordinate system is shown in Figure 1. Equation (1) can be written using the end point information as,

$$\mathbf{R} = (F_1(\xi)\ F_2(\xi)\ F_3(\xi)\ F_4(\xi))\ (\mathbf{R}(0)\ \mathbf{R}(1)\ \mathbf{R}_{,\xi}(0)\ \mathbf{R}_{,\xi}(1))^T \qquad (2)$$

---

[1] Numbers in square brackets represent references

where,

$$
\begin{aligned}
F_1(\xi) &= 2\xi^3 - 3\xi^2 + 1 \\
F_2(\xi) &= -2\xi^3 + 3\xi^2 \\
F_3(\xi) &= \xi^3 - 2\xi^2 + \xi \\
F_4(\xi) &= \xi^3 - \xi^2 .
\end{aligned}
\qquad (3)
$$

Equation (2) is referred to as the geometric form of the parametric cubic curve. [3] The geometric form of a line or curve provides a technique for determining the value of $\mathbf{R}(\xi)$ continuously over the curve with only knowing the starting and ending values and the starting and ending derivatives.

Surfaces are generated from 4 parametric curves and volumes may be defined by four surfaces spaced along a parametric coordinate. The parametric coordinate, ξ, is given a subscript to define the two and three dimensional geometric entities. A sketch of a parametric cubic surface and volume with the parametric coordinates are shown in Figure 2. In Figure 2, the sides of the surface and volume are shown as straight but they may also be curved. It is convenient to define edges on the surface and surfaces on the volume. Figure 2 also shows the edge and surface definitions. Clearly, edges could now be defined from the surfaces on the volume.

After the model has been defined, grids are assigned to the basic entities by specifying the number of grid points desired in a specific parametric direction. Consider the surface shown in Figure 3 which has been discretized with four grids in the ξ_1 direction and three grids in the ξ_2 direction. Recall that the parametric coordinates range in value from zero to one. The origin for the surface is identified by the arrows indicating the parametric directions. The one direction is uniformly divided into thirds to accommodate the requested number of grid points and the two direction is divided into halves. The parametric coordinates of the grid point indicated by the arrow in Figure 3 are (2/3, 1/2). The simple example shown in Figure 3 illustrates how the parametric coordinates are assigned and implies how a procedure could be developed for extracting grid point ID's from the parametric geometry. The grid point locations are stored in terms of their parametric coordinates. For example, in Figure 2, all grids on edge 3 would have parametric coordinates of $(1, \xi_2)$ and all grid points on surface 6 of the volume in Figure 2 would have parametric coordinates of $(\xi_1, \xi_2, 1)$. Search routines can be employed to located grid points with specific parametric coordinates based on the user input of which edge or surface information is needed. Next the program which makes use of the neutral file to extract grid point ID's will be discussed.

PROGRAM DESCRIPTION

A computer program was written in Fortran 77 which makes use of a neutral file for extracting the grid point ID's. Assume that a finite element model has been written to a neutral file and the user wishes to extract grid point ID's. Referring to Figure 4, the program will first ask for the name of the neutral file and also prompt the user for the name of a file where the extracted information is to be stored. Once the requested information has been entered, the program will respond with a summary of the model. Each line surface and volume is assigned an ID number and upon execution of the program these ID numbers appear in the output to the screen and are also written to the designated file. Thus, the user is informed about the content of the model as to the total number of grid points and the individual totals for the number of lines, surfaces and volumes. The user is asked if more information is wanted on lines, surfaces or volumes. If "none" is selected, the program will stop depending on wether a new model is to be investigated. The point "A" was designed into the program which will be convenient as a return location to request more information. If more information on lines, surfaces or volumes was requested, the program moves to location "B" on Figure 5.

The program flow chart shown in Figure 5 describes the procedure programed to extract grid point ID's from an individual line, surface or volume. Grid point ID's may also be obtained for the entire model. If a surface is selected, the user may also specify any edge on the surface. If a volume is selected, the user may also specify a surface on the volume. The program returns the user to point "A" on Figure 4 to request more information. The grid point ID's that were requested are written to an output file.

EXAMPLE PROBLEM

Consider a rather .simple model consisting of Line 31, Surface 12, and Volume 17 as shown in Figure 6. The model does not represent a real object but was designed to illustrate the utility of the grid point extraction program. The ID numbers for the geometric entities would be assigned by the preprocessor. Grid points have been assigned to each of the geometric entities and are also shown in Figure 6. The parametric coordinates are displayed to identify the edges and faces. Upon executing the program the analyst would enter the neutral file name containing the model and any name for an output file. The program will respond with the following summary:

```
Total No. of Grids:      23
No. of Lines:     1
No. of Surfaces   1
No. of Volumes    1
```

Suppose the objective is to recover grid point ID's on face 4 of Volume 17, (ie, grid points 18, 19, 22, and 23). These grid points can be identified by simply knowing which volume ID and which face. The face ID requires knowing the orientation of the parametric coordinate system of that particular volume. The preprocessor must provide this information graphically. Once the model summary has been written to the screen and output file, the program will prompt the user for more information on a L,S or V. The user will input "V" and the program will respond with:

```
Volumes  1
With the following ID's:
17
```

The next prompt asks for the ID number of the volume, in this case 17.

```
Volume 17  No. of Grids  8
16 17 18 19 20 21 22 23
```

The volume face ID is requested and the number 4 is input. Face 4 is located at the arrowhead end of the ξ_2 direction. The program responds with:

```
Face 4
Volume 17
18 19 22 23
```

The face request may be repeated. If no other faces are to be investigated, the program returns to the location "A" in Figure 4 which asks the user if any additional lines, surfaces or volumes are to be investigated.

CONCLUDING REMARKS

Most finite element preprocessors can identify a group of grid points, but there is no convenient method for writing this information to a NASTRAN set card. A program has been written to interrogate a neutral file representing a finite element model which over comes this deficiency. The program will extract grid point ID's which are associated with the parent geometry of the model. The program is external to the pre and post processor which provides additional capability for the analyst. The procedure makes use of the parametric description of lines, surfaces and volumes. The program is very useful for adding additional constraint or set cards to an existing model.

A logical extension to the work presented is to add the capability to define lists of grid point ID's that are not wholly contained in a geometric entity. It is often desirable to write constraint and data recovery sets that contain the grid point ID's from portions of several entities. These lists may then be used to write new constraint and data recovery sets without modifying the basic model geometry. Boolean operators may be used to generate the new sets.

REFERENCES

1. Ross, B.A., J.J. Cox, T.N. Hurst, S.E. Hurst, "Rosetta Stone For FE Modeling," Computers In Mechanical Engineering, Vol 3, No. 4, January 1985, pp 43-51.

2. Stanton, E.L., L.M. Crain, T.F. Neu, "A Parametric Cubic Modelling System General Solids of Composite Materials," International Journal For Numerical Methods in Engineering, Vol. 11, 1977, pp 653-670.

3. PDA Engineering, PATRAN USERS MANUAL,Vol 2, Release 1.5 Chapter 28.0, 2975 Redhill Avenue, Costa Mesa, Ca. 92626

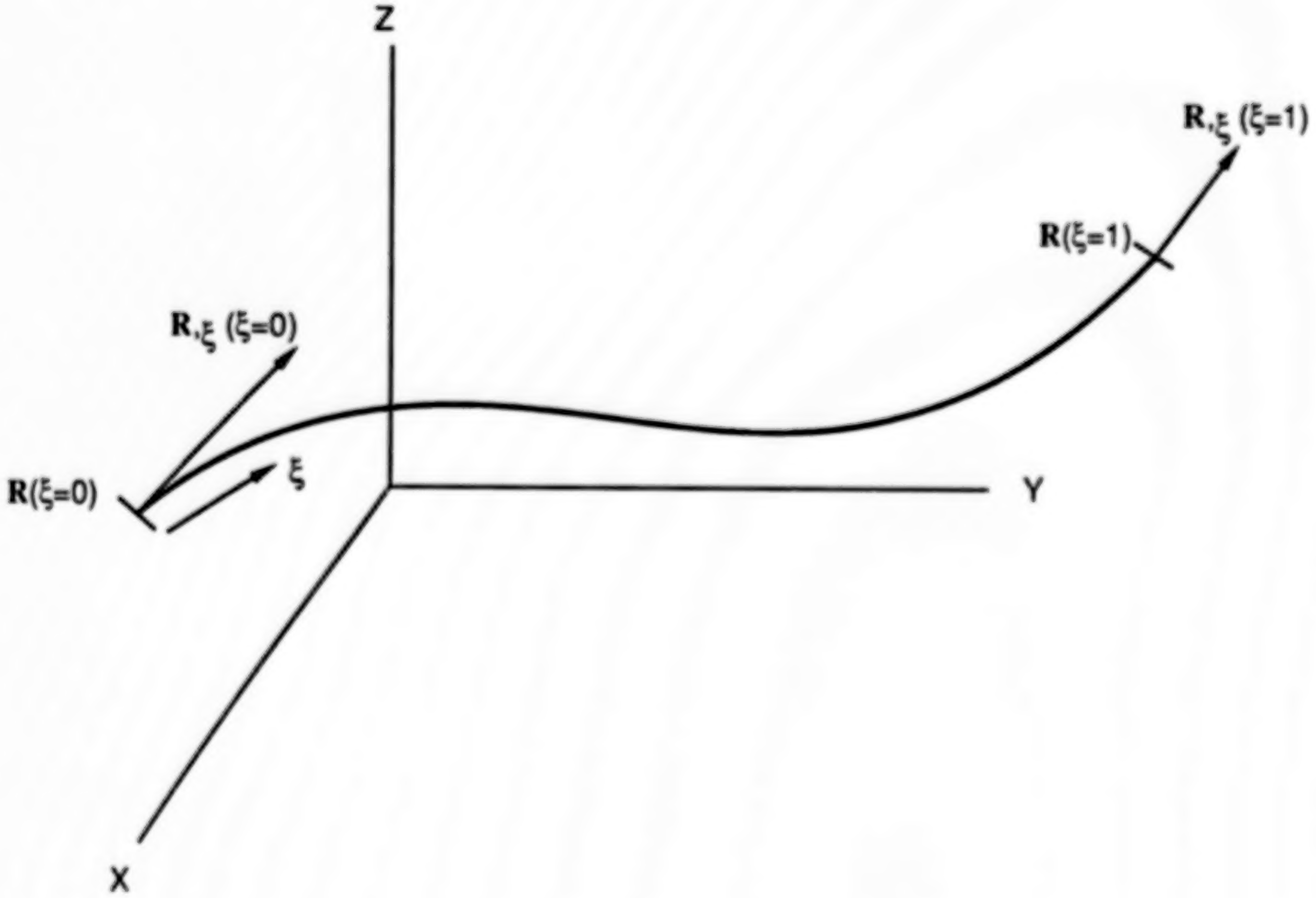

Z
R,ξ (ξ=1)
R(ξ=1)
R,ξ (ξ=0)
R(ξ=0)
ξ
Y
X

Figure 1 A Parametric Cubic Curve in a Cartesian Coordinate

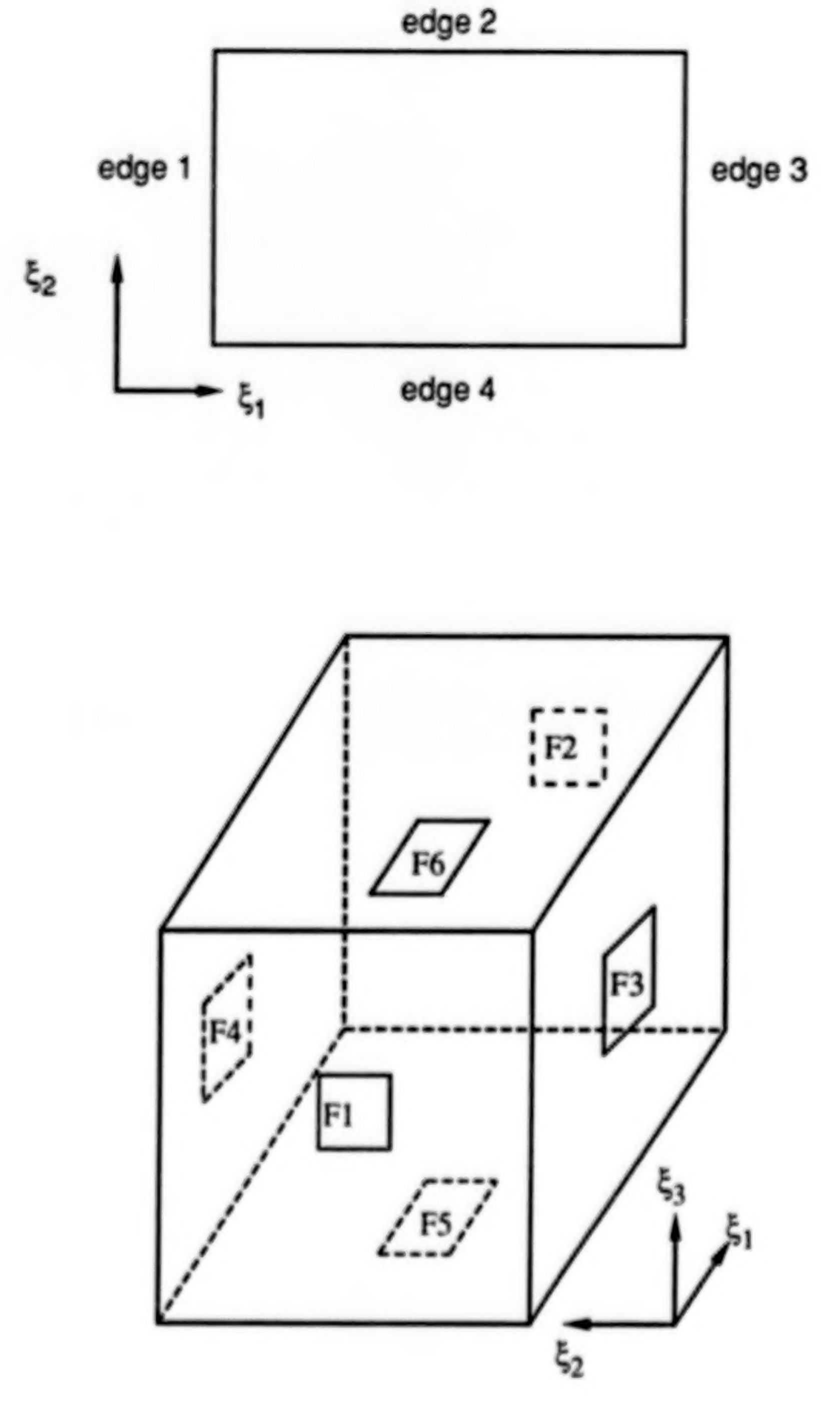

edge 2
edge 1
edge 3
ξ_2
ξ_1
edge 4
F2
F6
F3
F4
F1
F5
ξ_3
ξ_1
ξ_2

Figure 2. The Parametric Description of A Surface and Volume

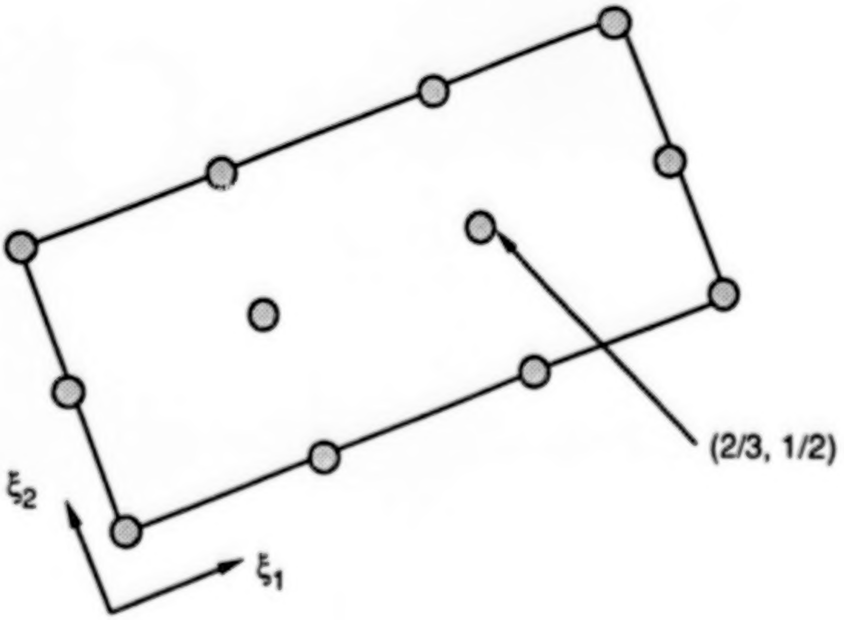

(2/3, 1/2)
ξ_2
ξ_1

Figure 3. Grid Points on a Surface Located in Parametric Coordinates

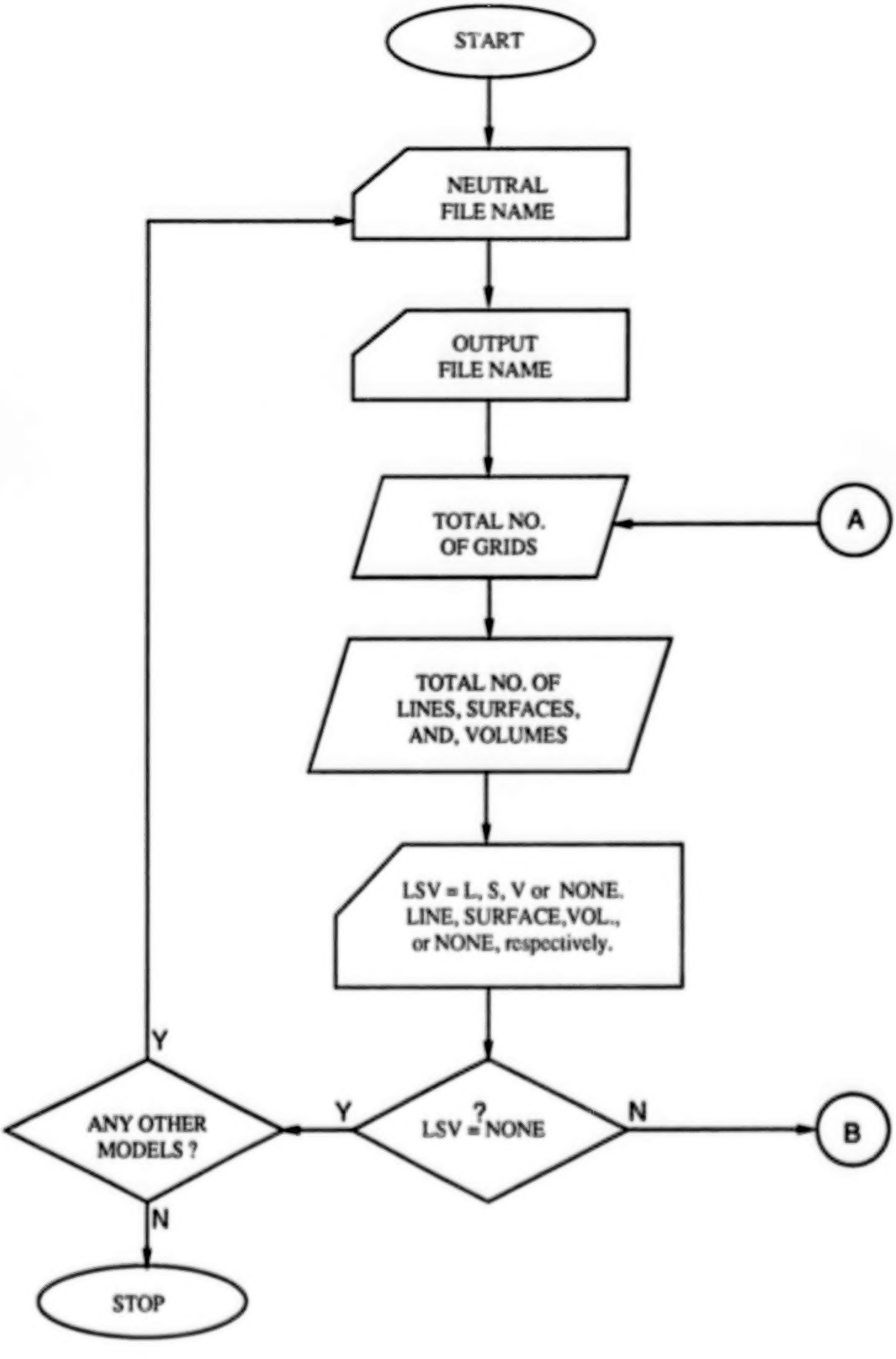

START
NEUTRAL FILE NAME
OUTPUT FILE NAME
TOTAL NO. OF GRIDS
A
TOTAL NO. OF LINES, SURFACES, AND, VOLUMES
LSV = L, S, V or NONE. LINE, SURFACE,VOL., or NONE, respectively.
LSV ?= NONE
Y
N
B
ANY OTHER MODELS ?
Y
N
STOP

Figure 4 Program Flow Chart for General Model Information

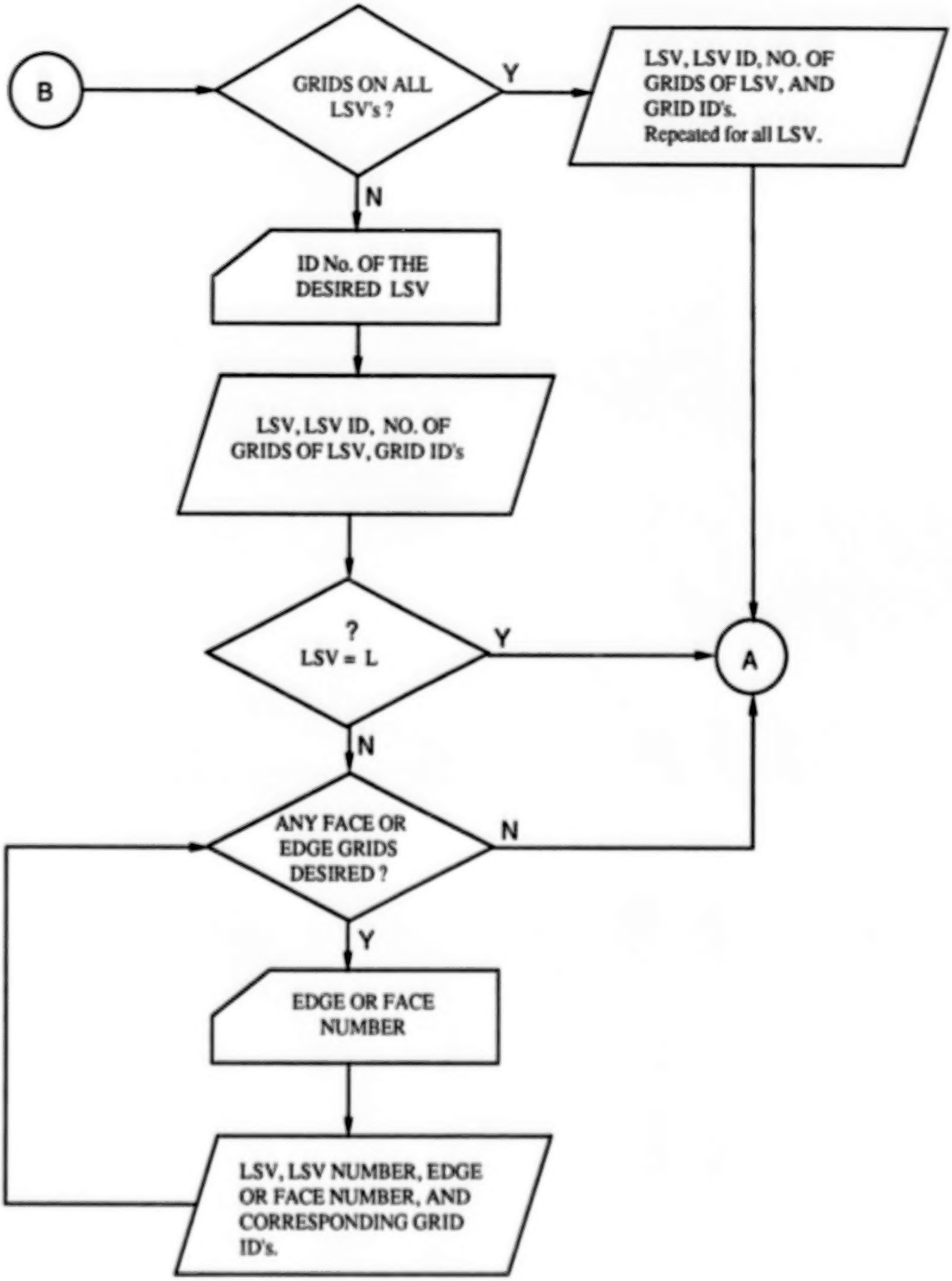

B
GRIDS ON ALL LSV's ?
Y
LSV, LSV ID, NO. OF GRIDS OF LSV, AND GRID ID's. Repeated for all LSV.
N
ID No. OF THE DESIRED LSV
LSV, LSV ID, NO. OF GRIDS OF LSV, GRID ID's
? LSV = L
Y
A
N
ANY FACE OR EDGE GRIDS DESIRED ?
N
Y
EDGE OR FACE NUMBER
LSV, LSV NUMBER, EDGE OR FACE NUMBER, AND CORRESPONDING GRID ID's.

Figure 5 Program Flow Chart for Grid Point ID Extraction

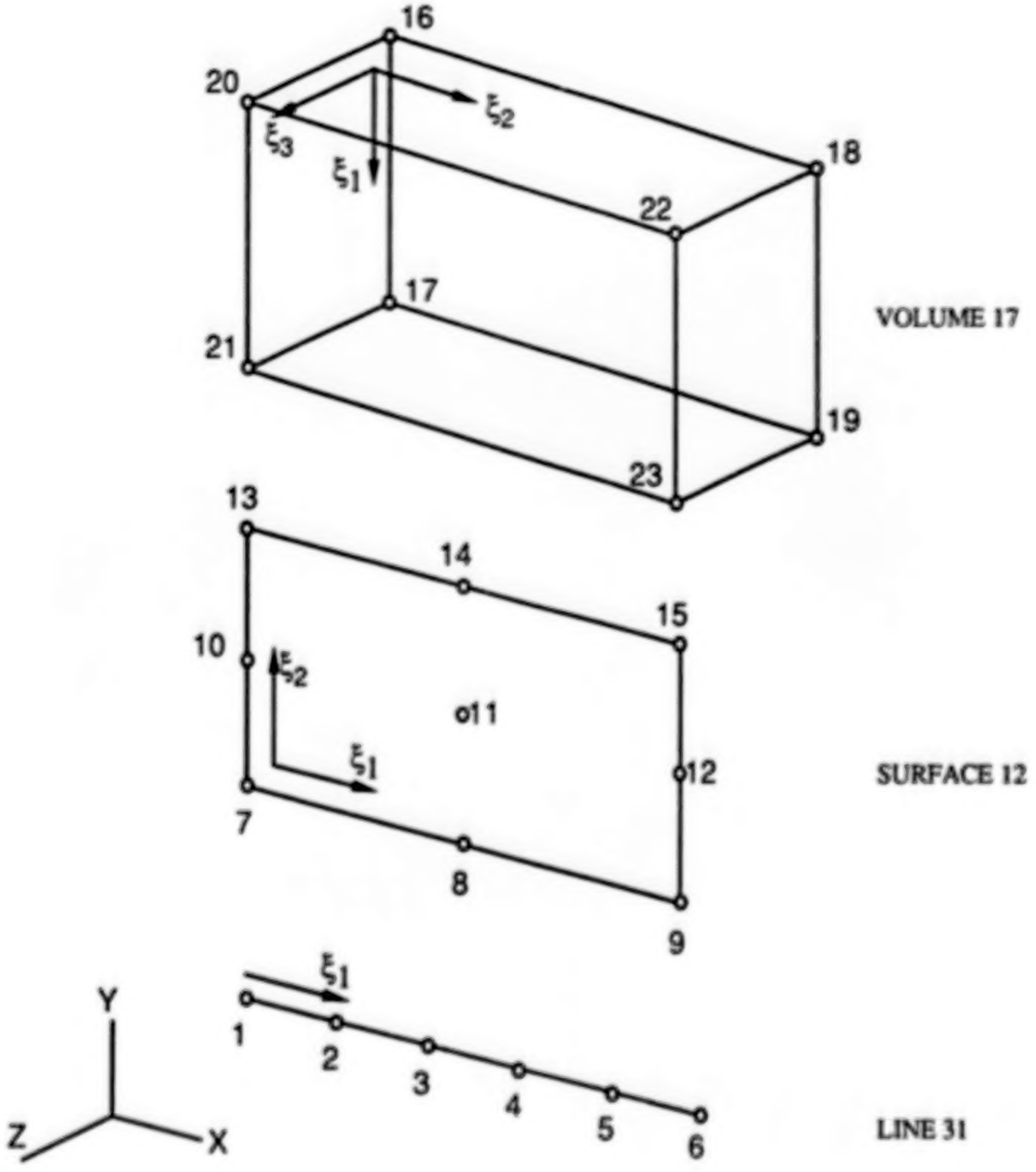

16
20
ξ_2
ξ_3
ξ_1
18
22
17
VOLUME 17
21
19
23
13
14
15
10
ξ_2
o11
ξ_1
12
SURFACE 12
7
8
9
ξ_1
Y
1
2
3
4
5
6
Z
X
LINE 31

Figure 6 - Line, Surface, Volume with Assigned Grid Points

1. Report No. NASA CP-3029	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle

Seventeenth NASTRAN® Users' Colloquium

5. Report Date

March 1989

6. Performing Organization Code

7. Author(s)

8. Performing Organization Report No.

10. Work Unit No.

9. Performing Organization Name and Address

COSMIC, NASA's Computer Software Management and
Information Center
University of Georgia
Athens, GA 30602

11. Contract or Grant No.

12. Sponsoring Agency Name and Address

National Aeronautics and Space Administration
Washington, DC 20546

13. Type of Report and Period Covered

Conference Publication

14. Sponsoring Agency Code

15. Supplementary Notes

Also available from COSMIC, Athens, GA 30602

16. Abstract

This document is the proceedings of a colloquium and contains technical papers contributed during the Seventeenth NASTRAN® Users' Colloquium held in San Antonio, Texas, April 24-28, 1989. The authors review general application of finite element methodology and the specific application of the NASA Structural Analysis System, NASTRAN, to a variety of static and dynamic structural problems.

17. Key Words (Suggested by Author(s))

NASTRAN
structural analysis
structures
finite element analysis
colloquium

18. Distribution Statement

Unclassified - Unlimited

Subject Category 39

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of pages 404	22. Price* A18

*For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA-Langley, 1989

National Aeronautics and
Space Administration
Code NTT-4

Washington, D.C.
20546-0001

Official Business
Penalty for Private Use, $300

National Aeronautics and
Space Administration

Washington, D.C.
20546

SPECIAL FOURTH CLASS MAIL
BOOK

Postage and Fees Paid
National Aeronautics and
Space Administration
NASA-451

Official Business
Penalty for Private Use $300

L1 001 SP-3029 [illegible]
GOVERNMENT PRINTING OFFICE
LIBRARY PROGRAMS SERVICE/SLLA
ACQUISITIONS SECTION
WASHINGTON DC 20401

POSTMASTER: **If Undeliverable (Section 158 Postal Manual) Do Not Return**

.396

END

DATE FILMED

12-19-91